Lean-periaatteiden mukaisen hukan havaitseminen ja minimointi ohjelmisto-organisaatiossa: tapaustutkimus

Riku Heinonen

Helsinki 16.11.2015
Pro Gradu -tutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Riku Heinonen

Työn nimi – Arbetets titel – Title

Lean-periaatteiden mukaisen hukan havaitseminen ja minimointi ohjelmisto-organisaatiossa: tapaustutkimus

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro Gradu -tutkielma	16.11.2015	89

Tiivistelmä – Referat – Abstract

Tämän tutkielman tarkoituksena on tutkia, kuinka kohteena olevasta ohjelmistokehitysprojektista voidaan havainnoida hukkaa, minkä tyyppisiä ja kuinka paljon mitäkin hukkaa havaitaan ja kuinka näitä hukkia voidaan poistaa. Tutkielmassa esitellään myös yksi lähestymistapa hukan lähteiden poistamisen priorisointia varten.

Tutkielmassa alussa tutustutaan Lean-ajatteluun yleisesti ja tarkemmin sen soveltamiseen ohjelmistokehityksessä. Erityisenä mielenkiinnon kohteena on Lean määritysten mukainen hukka sekä kuinka sitä voidaan havaita ja poistaa ohjelmistokehitysprosessista. Tutkielmassa myös esitellään käytetyt tutkimusmenetelmät, menetelmien tulokset ja analysoidaan niitä.

Tarkastelun kohteena oli erään valtiollisen toimijan IT-yksikön ohjelmistokehitysprojekti, joka alkoi joulukuussa 2011 ja päättyi vuoden 2015 maaliskuussa. Projektissa käytettiin monia ketteriä kehitysmenetelmiä, mutta ei Lean-menetelmiä suoranaisesti. Tutkielman aineisto kerättiin kartoittamalla arvovirtoja sekä muistioiden, raporttien, haastatteluiden ja kyselyn sisältöanalyysillä.

Havaitut hukan lähteet jaoteltiin ohjelmistokehitykseen sovellettuihin 7 hukan luokkaan ja niiden aiheuttamasta hukasta pyrittiin keräämään mahdollisimman paljon vertailukelpoista tietoa. Luokkien mitattua hukkaa myös verrattiin koettuun hukkaan, jota mitattiin kyselyllä. Projektista pystyttiin havainnoimaan jokaisen luokan mukaista hukkaa. Suurimmasta osasta saatiin kuitenkin vain viitteellistä tietoa, eikä näiden määrää tai vaikutusta pystytty projektista kerätyistä tiedoista osoittamaan. Ne hukat, jotka saatiin mitattua, olivat kyselylomakkeella kerätyn aineiston mukaan paljon hukkaa aiheuttavia. Vastavuoroisesti jotain merkittäviksi koettuja hukkia ei pystytty mitta-aineistosta havaitsemaan.

Kerättyjen tietojen perusteella voitiin tehdä selviä johtopäätöksiä hukista ja antaa suosituksia näiden poistamiseksi tai niiden vaikutuksen vähentämiseksi. Jatkotutkimuksen kohteiksi tunnistettiin esimerkiksi annettujen suositusten vaikutuksen tarkasteleminen.

ACM Computing Classification System (CCS): D.2 [Software Engineering]

Avainsanat – Nyckelord – Keywords

Lean, hukka, ohjelmistokehitys

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

SISÄLTÖ

1 Johdanto

Ohjelmistoja kehitetään usein tehostamaan jonkin yrityksen tuotantoa tai täyttämään jokin tarve kuluttajamarkkinoilla. Yritysten ja kuluttajien tarpeet kuitenkin muuttuvat yhä kovenevalla tahdilla. Näin ollen ohjelmistotuotannon pitäisi myös pystyä vastaamaan tarpeisiin yhä tehokkaammin. Esimerkiksi ketterät kehitysmenetelmät ovat auttaneet ohjelmistotuottajia tehostamaan toimintaansa [DyD08]. Nämä menetelmät eivät kuitenkaan ole se hopealuoti, joka ratkaisee kaikki ongelmat. Yhä suuri osa ohjelmistokehitysprojekteista myöhästyy tai ylittää budjettinsa [CaG12].

Tästä syystä Lean herättää mielenkiintoa ohjelmistoalalla. Lean tarkoittaa suoraan suomennettuna laihaa, hoikkaa tai niukkaa. Leanin tarkoituksena on tuottaa mahdollisimman paljon asiakkaan määrittelemää arvoa mahdollisimman vähillä resursseilla, olivat resurssit sitten aikaa, ihmisiä tai komponentteja. Tähän päästään, kun tuotetaan täsmälleen haluttuja tuotteita juuri oikeaan aikaan, tuotannon häiriöt korjataan välittömästi ja poistamalla tuotannosta arvoa tuottamatonta toimintaa eli hukkaa.

Leanin juuret ovat syvällä japanilaisessa autoteollisuudessa, josta sen ensimmäinen formaali ilmentymä nousi esiin nimellä Toyota Production System, ja sen periaatteiden avulla on pystytty saavuttamaan merkittäviä parannuksia tuottavuudessa verrattuna perinteisiin tuotantomenetelmiin [Had12]. Tästä järjestelmästä Womack ja Jones [WoJ98] yleistivät länsimaissa paremmin tunnetun Lean-filosofian.

Autoteollisuudessa havaittu tuotannon selvä tehostuminen on saanut myös ohjelmistokehittäjät kiinnostumaan Leanin soveltamisesta omiin tarkoituksiinsa. Ohjelmistokehitykseen liittyvät menetelmät ja prosessit ovat kuitenkin luonteeltaan merkittävästi erilaisia verrattuna autoteollisuuteen. Autoteollisuudessa suuri osa työstä on standardoitua ja toistavaa. Ohjelmistokehitys taas on uuden tuotteen kehittämistä ja tuotantoprosessiin liittyy usein jatkuvia muutoksia. Ramanin [Ram98] mukaan Leanin periaatteet ovat kuitenkin sovellettavissa myös ohjelmistokehitykseen. Poppendieckit [Pop03] [Pop06] ovat soveltaneet alkuperäisiä periaatteita ohjelmistokehitykseen paremmin sopivaan muotoon.

Tässä tutkielmassa erityisen mielenkiinnon kohteena on hukan havainnointi ja sen poistaminen ohjelmistokehitysprosesseissa. Tutkielmassa on suoritettu tapaustutkimus erään keskisuuren valtiollisen toimijan (jäljempänä toimija) IT-yksikön ohjelmistoprojektissa. Organisaatiossa on jo otettu käyttöön ketteriä menetelmiä ja kokeilua Lean-

menetelmien hyödyntämisestä ohjelmistokehitysprosesseissa on aloitettu. Tarkastelun tavoitteena on pyrkiä havaitsemaan yksikön ohjelmistoprojekteissa esiintyvää hukkaa ja tarjoamaan toimintavaihtoehtoja hukan vähentämiseksi.

1.1 *Tutkimuskysymykset*

Tutkielman tavoitteena on analysoida toimijan organisaation sisäisen IT-osaston ja sen sidosryhmien toimintaa ohjelmistokehitysprosessien arvovirtauksen tehokkuuden näkökulmasta. Tarkoituksena on tunnistaa tutkimuskohteena olevasta IT-osaston ohjelmistokehitysprojektista Lean-periaatteiden mukaista hukkaa ja kategorisoida ne ohjelmistokehitykseen sovellettujen hukkaluokkien mukaisiin kategorioihin.

Kun hukkatekijät on kartoitettu, tavoitteena on löytää Lean-työkaluista keinoja, joilla nämä hukkatekijät voidaan poistaa, tai ainakin niiden vaikutusta vähentää, näin tehostaen tuotantoa. Hukkaa ja siihen liittyvää terminologiaa esitellään tarkemmin luvussa 2.

Tutkimustavoite, tutkimuskysymykset ja metriikka on esitelty Goal-Question-Metric-kehyksen mukaisesti taulukossa 1.

Tavoite	Parantaa ohjelmistokehityksen tehokkuutta kohdeorganisaaiton asiakkaan näkökulmasta
Kysymys 1	Mitkä olivat suurimmat hukkaa aiheuttaneet tekijät päättyneessä ohjelmistoprojekteissa?
Kysymys 2	Millä toimilla voidaan havaittujen hukkatekijöiden vaikutusta kohdeorganisaation projekteissa vähentää tai poistaa?
Apukysymys 1	Mitä hukka on ohjelmistokehityksessä ja miten sitä voidaan tunnistaa?
Apukysymys 2	Miten hukan lähteiden poistoa voidaan priorisoida?
Metriikka	Hukattujen työpäivien määrä, hukan koettu määrä.

Taulukko 1: Tutkimuskysymysten GQM-kehys [BaG94]

Tutkimuskysymyksiin vastaamiseen tarvittava ainestoa esitellään aliluvussa 4.2.. Kysymykseen 1 vastataan luvussa 5, kysymykseen 2 ja apukysymyksiin 1 ja 2 luvussa 6.

1.2 *Tutkielman rakenne*

Luku 1 on johdanto, jonka jälkeen luvussa kaksi esitellään Leanin kehityshistoriaa sekä sen perusperiaatteita. Kolmannessa luvussa esitellään tutkimuksen kohteena oleva organisaatio sekä sen tutkimushetkellä käyttämät ohjelmistokehitysmenetelmät. Luvussa 4 käydään läpi tutkielmassa käytetyt tutkimusaineistot sekä sovelletut tutkimusmenetelmät. Viidennessä luvussa esitellään aineistosta saadut havainnot ja analysoidaan niitä teoriataustaan vasten. Luku 6 sisältää tutkimustulosten pohjalta tehdyt päätelmät, tutkimuksen luotettavuuteen liittyen seikkojen pohdinnan sekä mahdolliset ratkaisut havaittujen hukkatekijöiden poistamiseksi. Seitsemännessä ja viimeisessä luvussa on tutkielman yhteenveto.

2 Leanin esittely

Lean termillä voidaan tarkoittaa sekä tuotantotapaa että tapaa ajatella. Tiivistettynä Leanin tarkoituksena on tuottaa mahdollisimman paljon arvoa asiakkaalle minimaalisella määrällä resursseja. Sen ydin rakentuu tiettyjen periaatteiden ympärille. Kun nämä periaatteet on ymmärretty ja sisäistetty, voidaan niiden pohjalta luoda melkein mihin tahansa ympäristöön sopivia käytänteitä.

Nimensä Lean sai John Krafcikin artikkelissa *Triumph of the Lean Production System* [Kra88]. Artikkelissa Toyotan ja General Motorsin yhteistehtaalla työskennellyt Krafick kuvasi sanalla Lean (suom. hoikka, niukka) sitä, kuinka vähän resursseja tehtaalla käytettiin verrattuna perinteiseen massatuotantoon. Leanin juuret ovatkin syvällä japanilaisessa autoteollisuudessa, erityisesti Toyotan yrityskulttuurissa. Sen alkulähteenä onkin Toyota Production System (TPS) [Lik04], jonka tavoitteena oli parantaa Toyotan tuottavuutta toisen maailmansodan jälkeisessä Japanissa, jossa resursseja ei aina yksinkertaisesti ollut saatavilla. TPS tunnetaan myös nimellä Lean-tuotantotapa.

Tässä luvussa esitellään aluksi hukan käsite, sillä se ja varsinkin sen poistaminen tuotantoprosesseista on kriittinen osa Leania. Hukkaan myös viitataan jatkuvasti hukan jälkeen esiteltävässä Toyota Production Systemissä sekä Lean-ajattelussa. Kappaleen lopuksi esitellään vielä Lean-periaatteiden soveltamista ohjelmistokehitykseen.

2.1 Hukan määritelmä Leanissä

Tässä aliluvussa vastataan taulukossa 1 esiteltyyn tutkielman apukysymykseen 1 ”***Mitä hukka on ohjelmistokehityksessä ja miten sitä voidaan tunnistaa?***”. Erittäin tärkeä osa Lean-tuotannon tehokkuuden kehittämisessä ja säilyttämisessä on hukan jatkuva havainnointi ja sen poistaminen arvovirrasta. Arvovirran määritelmä esitellään aliluvussa 2.3. Jotta hukkaa voidaan havaita ja poistaa, tulee ensin ymmärtää mistä hukka muodostuu.

Lean-termistössä puhutaan kolmenlaisesta, toisiinsa linkitetystä hukasta: muda, muri ja mura.

- **Muda** (*engl.* waste, *suom.* turhuus, ylimääräisyys tai hukka) tarkoittaa kaikkea lisäarvoa tuottamattoman työn, kuten tuotteiden ylimääräinen liikuttelu varastosta toiseen.
- **Muri** *engl.* overburden, *suom.* liian vaikea, liiallisuus tai väkisin tehty) tarkoittaa

resurssien ylikuormitusta, kuten ihmisten tai laitteiden pakottamista työskentelemään enemmän kuin ne pitkällä aikavälillä tehokkaasti kykenevät.

- **Mura** (*engl.* unevenness, *suom.* epätasaisuus, epätasalaatuisuus) tarkoittaa epätasaisuutta ja heiluntaa, esimerkiksi tuotantolinjalla on kaksi viikkoa kuukaudesta hiljaista ja kaksi viimeistä viikkoa tehdään ylitöitä.

Likerin [Lik04] mukaan yleinen ongelma Leanissä implementoineissa on se, että niissä keskitytään yleensä vain mudaan ja sen poistamiseen. Hänen mielestään muut mu-termit ovat kuitenkin yhtä tärkeitä tehokkaan arvovirran luomisessa. Esimerkkinä tuotantoprosessi, jonka tuottavuudelle on asetettu kuukausitavoite. Kuun loppupuolella prosessista vastaavalla osastolla on valtava paine saavuttaa asetettu tavoite. Osastolla tehdään valtavasti ylitöitä, käytetään komponentteja ja tuotetaan tuotteita varastoon. Seuraavan kuun alussa prosessin tuottavuus laskee komponenttien puutteen ja jo varastossa olevien tuotteiden takia. Näin tuotannon epätasaisuus (mura) aiheuttaa prosessin ylikuormitusta (muri), joka johtaa hukkaan (muda).

Likerin [Lik04] mukaan Toyota Production Systemissä muda-hukkaa voidaan luokitella 7 eri luokkaan:

Varastot

Varastot muodostuvat kaikesta siitä, mitä ei tarvita juuri nyt tilattujen tuotteiden tuottamisessa. Esimerkiksi keskeneräiset tuotteet sekä valmiit tuotteet, joita ei ole toimitettu asiakkaalle, ovat varastoja. Varastot johtavat muuhun hukkaan, kuten kuljetukseen ja ylimääräiseen käsittelyyn. Varastoilla voidaan myös tasata tuotannossa olevia ongelmia, näin piilottaen hukkaa sen sijaan että ne korjattaisiin.

Ylituotanto

Ylituotannolla tarkoitetaan tuotteiden tuottamista, joille ei juuri nyt ole tarvetta. Ylituotanto johtaa muihin hukkiin kuten varastointiin ja kuljetukseen.

Ylimääräinen käsittely

Ylimääräisellä käsittelyllä tarkoitetaan kaikkia toimenpiteitä, jotka eivät ole välttämättömiä tuotteen toimittamiseksi asiakkaalle, kuten tarpeettomat työvaiheet, vioista johtuvat uudelleentekeminen tai liian laadukkaan tuotteen tekeminen kun vähempi olisi riittänyt.

Kuljetus

Kuljetus viittaa materiaalien ja työn tarpeettomaan siirtämiseen esimerkiksi varastosta työpisteelle tai työpisteeltä toiselle. Kuljetusta tulisi minimoida, sillä se on arvoa tuottamatonta ja tuotteet voivat vahingoittua kuljetuksen aikana.

Liike

Liikkeellä tarkoitetaan kaikkia ylimääräisiä liikkeitä, joita työntekijät joutuvat tekemään johtuen työpisteiden huonosta suunnitellusta, varastoista tai ylituotannosta.

Odottelu

Jokin prosessin osa voi odottaa osia, tarvittavat osat odottavat kuljetusta, prosessin seuraava osa odottaa edellisen osan tuotoksia, ja kaikki odotus on arvoa tuottamatonta toimintaa.

Viat

Viat aiheuttavat uudelleentekemistä, vikojen korjaamista ja vikojen tarkastusta. Kaikki näiden voidaan katsoa olevan arvoa tuottamatonta toimintaa. Tuotteisiin jäänet viat myös aiheuttavat tyytymättömyyttä asiakkaissa.

2.1.1 Hukka sovellettuna ohjelmistotuotantoon

Kuten muutkin osat TPS:stä, ovat aliluvussa 2.1. esitellyt 7 muda-hukkaa parhaiten sovellettavissa perinteisessä teollisuudessa, kuten autoteollisuudessa. Tyypillisesti tämän kaltaiseen teollisuuteen liittyvä työ on pitkälle standardoitua ja toistavaa. Ohjelmistokehitys taas on suurelta osin uusien tuotteiden kehittämistä ja näin ollen eivät samat hukan määritelmät kaikilta osin istu yhtä hyvin kumpaankin toimintatapaan. Poppendieckit [Pop03] määrittelevätkin ohjelmistokehityksessä esiintyvät hukat seuraavasti:

Osittain tehty työ

Kaikki koodi, jota ei ole testattu, dokumentoitu ja otettu käyttöön tuotannossa on osittain tehtyä työtä. Esimerkiksi valmiit vaatimukset joita ei ole otettu toteutukseen, integroimaton koodi, testaamaton koodi, dokumentoimaton koodi sekä käyttöönottamaton koodi ovat kaikki osittain tehtyä työtä.

Osittain tehty työ voidaan suoraan rinnastaa tuotteiden varastointiin. Se sitoo resursseja, ja esimerkiksi testaamattoman koodin toimivuudesta ei ole mitään takeita. Mikä vielä pahempaa, osittain tehty työ saattaa sisältää tiedostamattomia riskejä, jotka haittaavat tai

estävät muuta työtä. Näin ollen siihen käytetty työ voi olla täyttä hukkaa, eikä asiasta ole varmuutta ennen kuin työ on tehty loppuun asti.

Ylimääräiset ominaisuudet

Ylimääräiset ominaisuudet ovat myös osa resursseja sitovaa varastointia. Suurta osaa ohjelmistojen ominaisuuksia käytetään harvoin tai ei ollenkaan. Ominaisuudet kuitenkin kasvattavat järjestelmän koodikantaa ja sitä kautta tuottaa lisää ylläpidettävää sekä lisää koodikannan monimutkaisuutta. Ylimääräisten ominaisuuksien aiheuttavat siis hukkaa myös niihin suoraan hukattujen työtuntien jälkeenkin, ja edustavatkin näin pahimmanlaatuista hukkaa.

Uudelleenopettelu

Kuten nimi kertoo, uudelleenopettelulla tarkoitetaan jo opitun tiedon unohtamista ja oppimista uudelleen. Keskimääräinen ihmismieli pystyy pitämään noin 7 asiaa kerrallaan aktiivisesti käsittelyssä [Mil56]. Koska ohjelmistokehitykseen liittyy usein huomattavasti enemmän kuin 7 muistettavaa asiaa, on Poppendieckien mukaan oleellista, että muistin tukena käytetään järjestelmiä, joihin opittua tietoa voidaan kerätä. Muuten esimerkiksi jo kokeiltuja ratkaisuja saatetaan kokeilla yhä uudestaan, johtaen hukkaan.

Siirrot

Siirrot tarkoittavat tässä yhteydessä tietämyksen siirtoa, esimerkiksi osittain koodatun komponentin antamista kollegan työstettäväksi ja kaiken työn jatkamiseksi tarvittavan tiedon välitystä henkilöltä toiselle. Iso osa ohjelmistokehitystyöhön liittyvästä tiedosta on implisiittistä ja parhainkaan dokumentaatio ei kata kaikkea koodiin liittyvää tietämystä. Koodia tuottaessa koodaaja tekee useita ratkaisuja perustuen konventioihin ja standardeihin. Hän myös tekee ratkaisuja perustuen moneen muuhun asiaan, kuten koulutukseen, osaamiseen ja vaikkapa kuinka aurinkoista ulkona juuri sillä hetkellä sattuu olemaan. Kaikki tämä luo implisiittistä tietämystä tuotoksesta, jota yleisesti on vaikeaa siirtää ihmisten välillä, ja josta osa yksinkertaisesti katoaa hyvin nopeasti.

Poppendieckien [Pop06] varovaisen arvion mukaan 50 % tietämyksestä katoaa yhden siirron aikana. Oletetaan että määrittelijä tekee vaatimusmäärittelyn ja määrittelijän implisiittinen tietämys sekä dokumentti itsessään sisältävät 100 % tiettyyn asiaan liittyvästä tietämyksestä. Vaatimusmäärittelijä siirtää sen ensimmäiselle kehittäjälle, joka aloittaa toteutuksen. Ensimmäinen kehittäjä siirtää määrittelyn ja toteutuksen toiselle kehittäjälle, joka jatkaa työtä. Pelkästään kahden siirtymän jälkeen alkuperäisestä tietämyk-

sestä on jäljellä noin 25 %. Tämän kaltainen aukko tietämyksessä on riski toteutuksen oikeellisuuden kannalta, ja tiedon uudelleen hankkiminen on hukkaa. Tästä syystä turhia siirtoja tulisi välttää.

Tehtävien vaihtaminen

Useissa tutkimuksissa on todettu, että vaihtaminen yksinkertaistenkin tehtävien välillä rikkoo keskittymisen ja vaatii aivoilta orientoitumista uuteen tehtävään [Kal09]. Tämä korostuu erityisesti ohjelmistokehityksessä, jossa harva tehtävä on triviaali, ja tehtävän tehokas suorittaminen vaatii vahvaa keskittymistä [DeL13]. Tästä syystä esimerkiksi usean projektin työstäminen samaan aikaan aiheuttaa hukkaa.

Viivytykset

Viivytyksillä tarkoitetaan turhaa odottelua, jota saattaa syntyä monella tavalla. Tarvittava osaaminen on kiinni jossain muussa tehtävässä tai jokin tarvittava osa on juuttunut tulliin. Osalle viivytyksistä ei voi parhaalla tahdollakaan vaikuttaa eikä näitä lueta hukaksi. Kuitenkin esimerkiksi kehitystyössä tarvittavan informaation hankala saatavuus on hukan lähde. Jos tarvittavaa tietoa ei ole heti saatavilla, kehittäjän täytyy joko keskeyttää tehtävä ja hankkia tietoa, vaihtaa toiseen tehtävään tai edetä parhaan arvauksen varassa. Kaikki näistä vaihtoehdoista aiheuttavat hukkaa.

Viat

Vikojen aiheuttama hukka on aina sitä suurempi, mitä myöhäisemmässä vaiheessa vika havaitaan. Mikäli vikoja toistuvasti havaitaan viimeisissä hyväksymisvaiheissa, on itse prosessi vikaa. Koodin kattavan testaamisen tarkoitus onkin pelkän vikojen havaitsemisen sijaan estää niiden syntyminen kokonaan. Mikäli virhe havaitaan, tulisi testejä joko muokata tai luoda niin, että kyseinen virhe ei enää pääse ikinä toistumaan. Kattavaa ja hyvin toteutettua testipatteristoa toimii myös hyvin järjestelmän dokumentoinnissa, sillä testien pitäisi kertoa testaajalle miten järjestelmän odotetaan toimivan.

2.1.2 Hukkien yhteenveto

Taulukkoon 2 on kerätty yhteen aliluvussa 2.1.1. esiteltyjen ohjelmistokehitykseen sovellettujen hukkien havaitsemis- ja poistamiskeinoja.

Hukka	**Syy miksi hukkaa**	**Havaitsemiskeinoja**	**Poistamiskeinoja**
Osittain tehty työ	Koodin toimivuudesta ei takuita, voi sisältää riskejä	Koodia ei testattu, koodia ei otettu käyttöön	Tehtävä kerrallaan tehdään ilman keskeytyksiä loppuun asti
Ylimääräiset ominaisuudet	Kasvattaa koodikantaa, vaikeuttaa ylläpitoa, ei välttämättä käytetä koskaan	Ominaisuutta käytetään erittäin vähän tai ei ollenkaan	Ei tehdä ominaisuuksia "varmuuden vuoksi", tarpeiden kriittinen analyysi
Uudelleenopettelu	Samoja asioita opetellaan uudestaan ja uudestaan	Samoja tietoja kerrataan jatkuvasti	Tiedon dokumentointi, automaattitestaus, ongelmien ratkonta yhteistyönä, tehtävä kerrallaan tehdään ilman keskeytyksiä loppuun asti
Siirrot	Kaikkea tietoa ei saada dokumentoitua, joten siirroissa aina katoaa tietoa	Tehtäviä siirretään tekijältä toiselle	Minimoi siirrot, moniosaava tiimi, dokumentoinnin parantaminen
Tehtävien vaihtaminen	Tehtävän vaihtaminen rikkoo keskittymisen, aiheuttaa uudelleenopettelua	Samalla henkilöllä useita, päällekkäisiä tehtäviä joita työstetään samaan aikaan	Tehtävä kerrallaan tehdään ilman keskeytyksiä loppuun asti, varmista resurssien saatavuus
Viivytykset	Turha odottelu ei tuota minkäänlaista arvoa	Tehtävät keskeytyvät pitkiksi ajoiksi rerussien tai tiedon puutteen vuoksi	Varmista resurssien saatavuus, lyhyet iteraatiot, tehokas kommunikointi
Viat	Vikojen korjaamiseen kuluu aikaa	Virheraportit	Automaattitestaus, jatkuva integraatio

Taulukko 2: Hukkien havaitsemis- ja poistamiskeinojen yhteenveto

2.2 Toyota Production System

Toyota Production System (TPS) [Lik04] on Leanin ensimmäinen ja tunnetuin ilmentymä. Tässä aliluvussa käsitellään sen syntyhistoriaa sekä sen peruskäsitteitä. TPS on myös monen myöhemmin kehitetyn Lean-järjestelmän perusta. Monet näistä järjestelmistä käyttävät samoja periaatteita ja työkaluja kuin se. Tästä syystä näiden ensimmäisten perusperiaatteiden esittely auttaa ymmärtämään myöhempiä sovelluksia sekä selittämään mitä Lean käytännössä on.

2.2.1 Toyota Production System ja Leanin syntyhistoria

Vuonna 1927 Toyoda Automatic Loom Works esitteli tekstiilialan insinööreille uusia automaattisia kutomakoneitaan. Kutomakoneet olivat erittäin tehokkaita ja niin pitkälle automatisoituja, että 20 työntekijää pystyi pitämään 520 käynnissä yötä päivää. Kyseisten koneiden kehitystyö oli vaikea ja pitkä prosessi, joka antoi alkusysäyksen TPS:n syntyyn.

Sakechi Toyoda rakensi ensimmäisen kutomakoneensa 1896 ja lisäsi siihen automaattisen puolanvaihtajan 1903. Laitteet valmistettiin yksitellen, käytännössä käsityönä. Laite oli kilpailijoihin verrattuna monimutkainen, ymmärrettävästi hidas valmistaa ja vaikeasti ylläpidettävä. Parantaakseen tuotantonsa laatua, Toyoda palkkasi amerikkalaisen insinöörin Charles A. Francisin, jonka tarkoituksena oli tuoda yhtiöön Eli Whitneyn kehittelemä ”amerikkalainen tuotantomalli”. Tämän mallin mukaisesti työntekijät tuottivat koneistetuilla työkaluilla standardoituja osia, jotka koottiin valmiiksi tuotteiksi. Tämä muutoksen ansiosta Toyoda valmisti parannellun kutomakoneen, joka myi erittäin hyvin ympäri maailman.

Yhdysvalloissa tämä tuotantomalli kehittyi Henry Fordin ansiosta käyttämään vaihdettavien, standardoitujen osien lisäksi vaihdettavia työntekijöitä, jotka tekivät tiukasti standardoituja tehtäviä. Tehtävät oli pilkottu niin yksinkertaisiin palasiin, että uusi työntekijä pystyttiin kouluttamaan erittäin nopeasti ja hänen suoritustaan pystyttiin mittaamaan minuuttien tarkkuudella. Toyodan tapauksessa osien ja tuotantolinjan standardoinnista huolimatta kutomakoneiden valmistus oli kuitenkin edelleen erittäin vaativaa ja Toyodalla olikin tapana palkata parhaat insinöörit Japanin yliopistoista. Hän myös piti kiinni työntekijöistään perustaessaan uusia yrityksiä. Toyodan käytti siis hyväkseen vaihdettavia osia ja välineitä, mutta piti kiinni ihmisistä.

Vuonna 1924 Sakichi Toyoda, yhdessä poikansa Kiichiron kanssa, patentoivat uuden

tyyppisen automaattisen kutomakoneen, joka oli valtava menestys. Koneista saaduilla myyntivoitoilla Kiichiro käynnisti autojen tuotantoon keskittyneen osasto yhtiön sisällä ja lähti Yhdysvaltoihin tutustumaan auton valmistuksen saloihin. Osasto valmisti ensimmäisen autonsa 1936 ja vuonna 1937 Toyota Motor Company perustettiin omaksi, erilliseksi yhtiökseen. Toinen maailmansota kuitenkin keskeytti tuotannon.

Toisen maailmansodan jälkeen Kiichiro kannusti yhtiötään saavuttamaan amerikkalaisten kilpailijoiden etumatkan, mutta sodanjälkeisen Japanin taloudellinen tilanne ei yksinkertaisesti ollut yhteensopiva yhtiön käyttämään amerikkalaiseen tuotantomalliin. Materiaaleista oli pulaa, tilauskannat pieniä ja tuotteilta haettiin enemmänkin vaihtelua kuin yhdenmukaista laatua. Hän visioikin tuotantotavan, jossa tuotantoon tarvittavat osat saapuisivat aina oikeaan paikkaan oikeaan aikaan, ”Just-in-Time”.

Tämän vision toteutuminen oli pitkä prosessi. Yhtiön koneistusosaston päällikön Taiichi Ohno tutki sekä Fordin massatuotantomallia, että amerikkalaisten supermarkettien tapaa hallita inventaarioitaan ja yhdisti tähän kokonaisuuteen oman tietämyksensä kudonnassa käytetyistä työvaiheista. Usean vuoden kokeilujen jälkeen Kiichiro Toyodan visio näki päivänvalon TPS tuotantomallina, kymmenen vuotta Kiichiron kuoleman jälkeen, vuonna 1962.

2.2.2 Toyota Production System ja sen perusteet

Eräs tunnetuimmista TPS:n kuvauksista on Jeffrey K. Likerin kirjassa *The Toyota Way* [Lik04]. Tässä kirjassa Liker esittelee TPS:n ydin periaatteet ja jakaa ne neljää kategoriaan: pitkän tähtäimen filosofia (Long-Term Philosophy), oikeat prosessit tuottavat oikeita tuloksia (The Right Process Will Produce the Right Results), lisää organisaation arvoa kehittämällä ihmisiä ja yhteistyökumppaneita (Add Value to the Organization by Developing Your People and Partners) sekä jatkuva juuriongelmien ratkaiseminen ajaa organisaatiotason oppimista (Continuosly Solving Root Problems Drives Organizational Learning).

2.2.2.1 Pitkän tähtäimen filosofia

Tähän kategoriaan kuuluu vain yksi, mutta sitäkin tärkeämpi periaate: perusta päätökset pitkän tähtäimen filosofiaan, jopa lyhyen tähtäimen taloudellisten tavoitteiden kustannuksella. Tällä Liker tarkoittaa, että yrityksellä tulisi olla pitkän tähtäimen filosofia: tapa toimia, jonka tarkoituksena on työskennellä ja kasvattaa organisaatiota kohti tavoitteita, jotka ovat merkittävämpiä kuin pelkkä rahanteko. Organisaation tulisi myös aina

kantaa vastuunsa kaikissa toimissaan.

Organisaation jäsenten tulisi myös ymmärtää oma paikkansa organisaation historiassa ja kaikkien tulisi pyrkiä ylläpitämällä sekä parantamalla arvoa tuottavia taitojaa viemään organisaatiota eteenpäin. Likerin mukaan tämä filosofia toimii perustana kaikille muille periaatteille.

2.2.2.2 Sopivat prosessit tuottavat aiottuja tuloksia

Tässä kategoriassa on seitsemän periaatetta:

- Luo prosesseista jatkuva virta nostaaksesi ongelmat pinnalle
- Käytä imuohjausta välttääksesi ylituotantoa
- Tasoita tuotannon heilunta
- Saadaksesi laadun kerralla oikein, rakenna kulttuuri joka pysähtyy korjaamaan ongelmat
- Standardoidut tehtävät ovat jatkuvan parantamisen ja työntekijöiden voimaannutamisen perusta
- Käytä visualisointeja paljastaaksesi ongelmia
- Käytä vain luotettavia sekä kattavasti testattuja teknologioita, jotka palvelevat työntekijöitäsi ja prosessejasi.

Luo prosesseista jatkuva virta nostaaksesi ongelmat pinnalle

Tämän periaatteen mukaisesti organisaation prosessit tulisi suunnitella uudestaan niin, että ne jatkuvana virtana tuottavat mahdollisimman paljon arvoa. Jatkuvalla virtauksella tarkoitetaan sitä, että kun tuotantoprosessi käynnistyy, materiaalit ja työtehtävät kulkevat sen läpi nopeasti ja ilman hukkaa, näin tehden tuotannosta tehokasta. Kun prosessit ja ihmiset on linkitetty jatkuvaksi virraksi, myös prosesseihin liittyvät ongelmat nousevat selvemmin esiin. Kun ongelmat on nostettu esiin, voidaan niitä ratkomalla sekä parantaa tuotantoa että kehittää ihmisten osaamista.

Likerin mukaan useimmissa prosesseissa 90 % on hukkaa ja 10 % on arvoa tuottavaa työtä. Hukkaa poistamalla tehostetaan arvon virtausta, mutta samalla muun muassa parannetaan tuotteiden laatua opettamalla kaikkia työntekijöitä tarkkailemaan työnsä jälkiä, parannetaan prosessien joustavuutta lyhentämällä kunkin työvaiheen läpimenoaikaa ja vapautetaan pääomaa vähentämällä tarvetta varastoinnille.

Käytä imuohjausta välttääksesi ylituotantoa

Imuohjauksen tarkoituksena on tuottaa asiakkaan haluamia tuotteita kulutuksen mukaan eli juuri oikeita määriä oikeaan aikaan. Perinteisesti markkinoiden kysynnän heilahteluihin on vastattu varastoimalla tuotteita. Näin kun kysyntää on, voidaan tuotteita toimittaa varastoista haluttu määrä, mikäli oikeita tuotteita on saatavilla.

Varastoihin liittyy kuitenkin monia ongelmia. Ne sitovat pääomaa valmiiden tuotteiden ja fyysisen säilytystilan muodossa, varastot saattavat täyttyä tarpeettomista tuotteista ja lopulta ne vanhenevat. Täydellisesti toimivassa imuohjatussa järjestelmässä varastot ovat hyvin pieniä tai niitä ei ole lainkaan.

Otetaan esimerkiksi autokorjaamo. Normaalisti korjaamolle toimitetaan tietty määrä vara-osia kerran kuussa oletetun kysynnän mukaan. Joinain kuukausina saattaa kuitenkin käydä niin, että osia kuluu oletettua vähemmän ja korjaamon varastot paisuvat uusien toimitusten myötä. Imuohjatussa järjestelmässä korjaamolla on vain muutamia osia varastossa. Kun osa käytetään, korjaamo tilaa välittömästi uuden ja se toimitetaan muutamassa päivässä.

Tasoita tuotannon heilunta

Mikäli tuotantomäärät vaihtelevat voimakkaasti kuukaudesta toiseen, ei imuohjausta voida tehokkaasti hyödyntää. Tuotannon heilahtelut myös rasittavat tuotannosta vastaavia ihmisiä ja työvälineitä, näin johtaen aiemmin esiteltyjen murin ja muran kautta mudaan eli hukkaan. Tuotanto tulisikin tasapainottaa niin, ettei voimakkaita suvantoja tai piikkejä synny. Tätä tasapainotusta kutsutaan heijunkaksi.

Heijunkassa tuotantoa tasapainotetaan sekä tuotantomäärien että tuotettavien tuotteiden mukaan. Tuotantoa muutetaan niin, että tuotteita tuotetaan keskimääräisten tilausten mukaan. Näin sekä materiaalien tarve että työn määrä pysyy tasaisena. Tämä saattaa synnyttää pieniä varastoja, mutta tämä on pieni hukka verrattuna kuinka paljon hukkaa voidaan poistaa käyttämällä imuohjausta.

Saadaksesi laadun kerralla oikein, rakenna kulttuuri joka pysähtyy korjaamaan ongelmat ja poistamaan niiden syyt

Tämän periaatteen mukaan laadun tulisi olla tärkein tekijä asiakkaille annetussa arvolupauksessa. Tähän päästään, kun laatua tarkkaillaan tuotannon jokaisessa vaiheessa. Kun missä tahansa vaiheessa ilmenee ongelma, tuotanto pysäytetään, ongelma korjataan välittömästi ja luodaan vastatoimet, jotta sama ongelma ei jatkossa toistu. Koska jokaisella

työntekijällä on valta ja vastuu näin tehdä, ongelmat ja virheet korjaantuvat tehokkaasti ja laatu paranee. Tätä toimintatapaa kutsutaan jidokaksi.

Jidoka tarkoittaa myös ”älykästä automaatiota” tai ”automaatiota ihmiskosketuksella”. Sen mukaan laitteet tulisi suunnitella pysähtymään automaattisesti, mikäli se havaitsee poikkeamia tai virheitä tuotannossa. Tästä varhaisena esimerkkinä Sakichi Toyodan luomat automaattikutimet, jotka pysähtyivät, mikäli koneessa oleva lanka katkesi. Näin ongelma voitiin havaita ja korjata nopeasti.

Tässä apuna käytetään myös andon-tekniikkaa. Sillä tarkoitetaan mekanismia, jonka avulla tuotanto pysäytetään ja signaloidaan, että tuotannossa on havaittu virhe. Toyotan autotehtaalla tämä mekanismi oli nappi, joka sytytti vilkkuvan valon, ilmoittaen millä asemalla ongelma havaittiin. Andon voi kuitenkin olla mikä tahansa visuaalinen väline (näyttö, valotaulu), jonka avulla signaali voidaan välittää.

Kun välitön ongelma on korjattu, seuraava askel on selvittää miksi ongelma syntyi ja miten se estetään jatkossa. Ongelman syytä selvitetään kysymällä viisi kertaa ”miksi?”. Useamman kysymyksen tarkoituksena on estää ongelmanselvitystä pysähtymästä ensimmäiseen, helposti vastaantulevaan vastaukseen. Kun ongelman juurisyy selvitetään mahdollisimman perusteellisesti, todennäköisesti ratkaistaan myös muita ongelmia jo ennalta.

Kun juurisyy on selvillä, voidaan sitä yrittää estää poka-yoke-välineillä, joiden tarkoituksena on estää virheitä tapahtumasta tai havaita tapahtunut virhe välittömästi. Poka-yoke voi olla esimerkiksi sensori, joka tarkistaa että tietty osa on asennettu tai työkalu, joka mahdollistaa ruuvien kiristämisen vain tiettyyn tiukkuuteen.

Tekniikoista ja apuvälineistä huolimatta, Likerin mukaan tärkeintä laadun takaamiseksi on rakentaa yrityskulttuuri, joka mahdollistaa jidokan käyttämisen.

Standardoidut tehtävät ovat jatkuvan parantamisen ja työntekijöiden voimaannutamisen perusta

Likerin mukaan standardoidut työtehtävät ovat tehokkaan arvovirran ja imujärjestelmän perusta. Ilman niitä prosessien parantaminen on mahdotonta, sillä mikäli työtehtävät tehdään aina eri tavalla, ongelmien juurisyytä on mahdoton selvittää.

Standardointi ei kuitenkaan tarkoita, etteikö työtehtäviä voitaisi muuttaa. Työtehtävät tulisi aluksi standardoida sen hetkisten parhaiden käytäntöjen mukaan, mutta kuitenkin sallia yksittäisten työntekijöiden parantaa niitä. Mikäli parannukset osoittautuvat toimi-

viksi, tulisi ne ottaa osaksi standardia ja tätä kautta levittää muille työntekijöille. Näin prosessit pysyvät ennustettavina ja tasaisina, mutta työntekijät kokevat voivansa vaikuttaa työhönsä.

Käytä visualisointeja paljastaaksesi ongelmia

Visualisointien tarkoituksena on mahdollistaa tuotannon tilanteen hahmottaminen yhdellä vilkaisulla. Visualisointien tulisi olla mahdollisimman yksinkertaisia ja niiden avulla pitäisi pystyä kertomaan noudattaako tuotanto asetettuja standardeja sekä ohjeita tai onko tuotannossa ongelmia. Visualisointien ei myöskään tulisi häiritä työntekoa.

Käytä vain luotettavaa, kattavasti testattua teknologiaa joka hyödyttää ihmisiä ja prosesseja

Teknologian tulisi tukea ihmisten toimintaa, ei korvata heitä. Yleensä parhaisiin tuloksiin päästään, kun prosessit käydään läpi manuaalisesti ennen teknologian lisäämistä siihen. Uusi teknologian on usein myös arvaamatonta ja saattaa haitata arvovirtausta. Tästä syystä toimiviksi todetut prosessit ovat arvokkaampia kuin uusi teknologia.

Henkilöstöä tulisi kuitenkin kannustaa tutkimaan uusien teknologioiden luomia mahdollisuuksia. Ne tulisi testata kattavasti ja tarvittaessa muokata sopimaan organisaation tarpeisiin. Mikäli uusi teknologia uhkaa prosessien jatkuvaa parantamista tai häiritsee ennustettavuutta, eikä sitä voida korjata muokkaamalla, tulisi se hylätä.

2.2.2.3 Paranna organisaation arvolupausta kehittämällä ihmisiä ja yhteistyökumppaneita

Tässä kategoriassa on kolme periaatetta:

- Kasvata johtajia, jotka ymmärtävät työn sisällön, elävät filosofian mukaisesti ja opettavat sitä muille
- Kehitä loistavia ihmisiä sekä tiimejä, jotka noudattavat yhtiösi filosofiaa
- Kunnioita laajennettua yhteistyö- ja toimittajaverkostoasi haastamalla heitä sekä auttamalla heitä parantamaan toimintaansa.

Kasvata johtajia, jotka ymmärtävät organisaation työtä, elävät sen filosofian mukaisesti ja opettavat sitä muille

Tämän periaatteen mukaan tulisi kasvattaa johtajia organisaation sisältä, eikä vain hankkia heitä sen ulkopuolelta. Johtajien tulisi toimia roolimalleina muille työntekijöille ja ymmärtää erittäin yksityiskohtaisesti organisaatiossa tehtävää työtä. Näin he voivat

parhaalla mahdollisella tavalla opettaa organisaation filosofiaa muille

Kehitä loistavia ihmisiä sekä tiimejä, jotka noudattavat organisaation filosofiaa

Organisaatioon tulisi luoda vahva ja vakaa kulttuuri, jonka arvot työntekijät jakavat. Kouluttamalla työntekijät ja tiimit toimimaan tämän filosofian mukaisesti voidaan saavuttaa erinomaisia tuloksia, ja siksi tätä kulttuuria tulisi vahvistaa mahdollisimman paljon.

Tuottavuutta ja laatua voidaan parantaa myös käyttämällä tiimejä, joihin kootaan monipuolista osaamista. Kun tiimit ratkovat vaikeita teknisiä ongelmia, organisaation arvovirrat tehostuvat ja tiimit voimaantuvat onnistumisten kautta. Tästä syystä tiimityö osaamisen tulisi olla korkealla prioriteetilla työntekijöiden koulutuksessa.

Kunnioita yhteistyö- ja toimittajaverkostoasi haastamalla heitä ja auttamalla heitä paranemaan

Tämä periaate korostaa sitä, että yhteistyö- ja toimittajaverkosto tulisi nähdä oman organisaation laajennuksena, ja näin ollen kohdella heitä samalla kunnioituksella kuin omaankin organisaatiota. Tätä verkostoa tulisi myös kehittää asettamalle heille haastavia tavoitteita ja auttamalla heitä saavuttamaan ne mahdollisimman tehokkaasti.

2.2.2.4 Juuriongelmien jatkuva ratkominen kehittää organisationaalista oppimista

Tähän kategoriaan Liker sijoittaa myös kolme periaatetta:

- Ymmärtääksesi tilanteen perusteellisesti, mene paikan päälle ja katso tilannetta itse
- Tee päätöksiä hitaasti yhteisymmärryksen kautta, tutkien kattavasti kaikki vaihtoehdot; kun päätös on tehty, jalkauta se nopeasti
- Muutu oppivaksi organisaatioksi jatkuvan itsetarkastelun ja toiminnan jatkuvan parantamisen kautta.

Ymmärtääksesi tilanteen perusteellisesti, mene paikan päälle ja katso tilannetta itse

Ongelmien ratkontaan ja prosessien parantamiseen käytettävä tietoa tulisi kerätä sekä varmentaa henkilökohtaisesti aina kun mahdollista. Johtajien kaikilla tasoilla tulisi itse tutustua ongelmiin saadakseen riittävän kattavan kuvan tilanteesta, eikä vain luottaa muiden välittämään tietoon.

Tee päätöksiä hitaasti yhteisymmärryksen kautta, tutkien kattavasti kaikki vaihtoehdot; kun päätös on tehty, jalkauta se nopeasti

Päätösten tulisi perustua useiden vaihtoehtojen tarkkaan harkintaan. Kun päätös on tehty, tulisi valittua polkua seurata nopeasti mutta varoen. Päätösten tulisi myös tapahtua yhteisymmärryksessä niiden osapuolten välillä, joihin päätös vaikuttaa. Tämä yhteisymmärrys saavutetaan jakamalla kaikki päätökseen tarvittava informaatio kaikille, jotka päätökseen osallistuvat ja kuuntelemalla kaikkien mielipiteitä. Tämän kaltainen prosessi saattaa hidastaa päätöksen tekoa, mutta kun päätös on saatu aikaiseksi, on sen toteuttaminen huomattavasti tehokkaampaa kun kaikki seisovat päätöksen takana.

Muutu oppivaksi organisaatioksi jatkuvan itsetarkastelun ja toiminnan jatkuvan parantamisen kautta

Kun organisaatio on saanut tietyn prosessin stabiloitua, tulisi sen käyttää jatkuvan parantamisen työkaluja tehottomuuden juurisyiden löytämiseen ja tehokkaiden vastatoimien luomiseksi. Organisaation tulisi myös kerryttää mahdollisimman paljon osaamista suorittamalla *hansei* eli itsetarkastelua kun saavutetaan tärkeitä virstanpylväitä. Näin voidaan avoimesti tunnistaa asioita, jotka toimivat hyvin ja joita kannattaa jatkaa, sekä tunnistaa asioita, joita voidaan parantaa. Hyviksi havaitut asiat tulisi standardoida, jotta samoja asioita ei aina tarvitsisi oppia uudelleen.

Organisaation tulisi myös suojella kerryttämäänsä tietoa ja taitoa panostamalla henkilökunnan vakiinnuttamiseen. Vaihtuvuus tulisi minimoida jopa hidastamalla työntekijöiden ylentämistä.

2.3 Lean Thinking

Tunnetuimpia TPS-johdannaisia on Womack ja Jonesin kirjassa *Lean Thinking* [WoJ98] esittelemä Lean-ajattelu. He pyrkivät yleistämään TPS:n periaatteita helpottaakseen Leanin soveltamista mille tahansa tuotantoalalle. Se perustuu viiteen ydinkonseptiin: asiakkaan määrittelemä arvo (value), arvovirta (value stream), arvovirran tasainen soljunta (flow), imuohjaus (pull) ja täydellisyys (perfection).

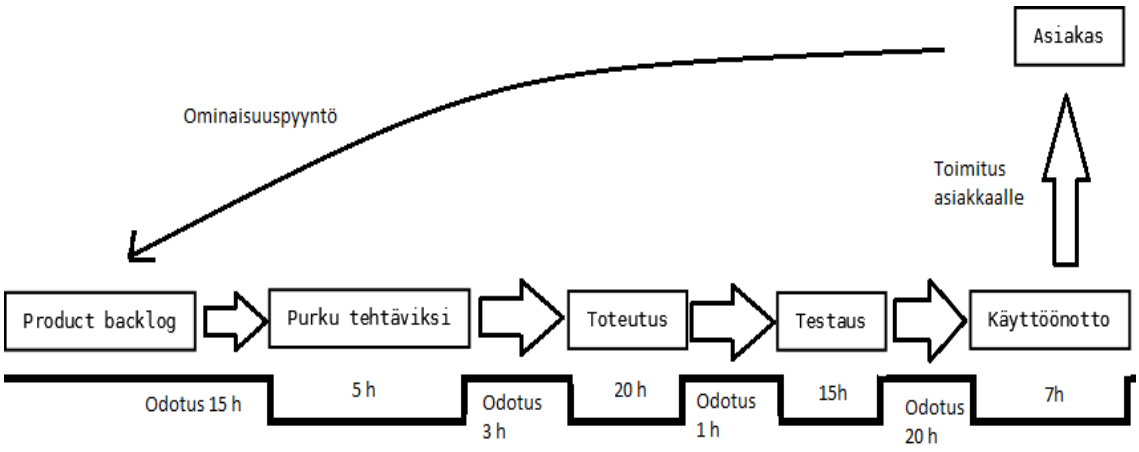
Arvon määrittely

Arvon määrittely on Lean-ajattelun kriittisin osa ja sen voi määritellä vain asiakas. Arvon tuottajien päämääränä on tuottaa asiakkaan kanssa yhteistyössä määritelty tuote, joka täyttää asiakkaan tarpeet. Kunkin toiminnan arvo voidaan siis määritellä kysymällä

”Onko tästä asiakkaalle hyötyä?”. Tähän viimeisen vastauksen antaa aina asiakas, ei tuottaja. Määritelmä on myös aina sidoksissa tiettyyn tuotteeseen ja tiettyyn ajanhetkeen, ilman tätä kontekstia määritelmä on merkityksetön. Jos jokin tuotantoprosessin vaihe auttaa tuottamaan asiakkaan määrittelemää arvoa, on se arvoa tuottavaa työtä (value adding). Mikäli se ei auta, on se arvoa tuottamatonta työtä (non-value adding), joka tulisi pyrkiä poistamaan prosessista. Tämän kaltainen työ voi kuitenkin olla myös välttämätöntä muun tuotantoprosessin kannalta, ja näin ollen sitä ei voida prosessista poistaa. Tätä kutsutaan välttämättömäksi, ei arvoa tuottavaksi työksi (necessary, non-value adding).

Arvovirta

Arvovirralla kuvataan kaikki tarvittavat vaiheet, jotka tarvitaan tietyn tuotteen tai palvelun tuottamiseksi asiakkaalle, ja sen tarkoituksena on tarkastella ja optimoida kokonaisuutta yksittäisten vaiheiden sijaan. Arvovirta aina alkaa ja päättyy asiakkaaseen.Tyypillisesti arvovirran määrittely paljastaa suuria määriä hukkaa eri vaiheissa. Arvovirta on erittäin oleellinen käsite, sillä järjestelmällinen hukan poistaminen ja arvovirran soljunnan jatkuva parantaminen ovat keskeinen osa Leaniä.



Kuva 1: Esimerkkikuva arvovirrasta

Olennainen osa arvovirran määritystä ovat käsitteet läpimenoaika (Lead Time) ja tahtiaika (Takt Time). Läpimenoajalla tarkoitetaan kokonaisaikaa, joka kuluu asiakkaan tilauksenteko hetkestä tilauksen toimittamiseen. Tahtiajalla tarkoitetaan aikaa, joka kuluu yhden tuotteen valmistamiseen. Arvovirran tehokkuutta tarkastellaan usein käyttämällä kuvan 2 mukaista arvovirtakarttaa, johon merkitään prosessin eri vaiheisiin käytetyt ajat sekä eri vaiheiden aloituksen odottamiseen käytetty aika. Tahtiaika saadaan laskemalla

yhteen eri työvaiheisiin käytetty aika, ja läpimenoaika laskemalla tähän summaan odotusajat.

Arvovirran tasainen virtaus

Arvovirran optimoinnin päämääränä on saada arvo virtaamaan keskeytyksettä. Kun ajatellaan vaikkapa tehokasta tapaa usean postipaketin lähettämiseksi, yleinen mielikuva on lähettämiseen liittyvien työtehtävien niputtaminen eriin tehtävän tyypin mukaan ja koko nipun suoritus kerralla. Ensin kaikki paketit pakataan, sitten se suljetaan, sitten laitetaan postiosoitteet ja näin edelleen. Vaikka tämän tyyppinen prosessointi on käytännössä toiminut jo vuosisatoja teollisen tuotannon pohjana, ei se välttämättä ole tehokkain tapa toimia.

Leanin pyrkimyksenä onkin niin sanottu *single piece flow*, jossa välineet ja tekijät on organisoitu niin, että yksittäinen tuote kulkee tilauksesta asiakkaan käteen ilman hukkatekijöitä kuten välivarastoja, ylimääräistä kuljetus tai ylimääräistä prosessointia. Tähän päästään poistamalla tuotantoprosessista hukkaa jatkuvasti.

Imuohjaus

Arvovirran liike-energia synnyttää painovoiman sijasta imu (pull). Siinä missä aiemmin tuotettiin varastot täyteen erilaista tavaraa odottamaan ostajia, voidaan Leanin avulla tuottaa juuri sellaisia tuotteita kun asiakas tarvitsee, juuri silloin kun niitä tarvitaan. Tarkoituksena on siis antaa asiakkaan tarpeen luoda imua järjestelmään eikä tehdä mitään ennen sitä. Imujärjestelmä tunnetaan myös nimellä *Just in time* -malli tai JIT.

Tämän vastakohtana toimii työntöön (push) perustava järjestelmä, jossa ennakoivasti valmistetaan tuotteita mahdollista tarvetta varten.

Täydellisyys

Kun arvovirta alkaa soljumaan, päästään hukkaa poistamalla lähemmäs sitä mitä asiakas oikeasti haluaa ja tarvitsee. Asiakas on tyytyväinen ja lisää tilauksia syntyy. Jatkuvasti vahvistuva arvovirta paljastaa kuitenkin jatkuvasti uusia hukkatekijöitä, eikä yksikään arvovirta ole ikinä täydellisen vapaa hukasta. Täydellisyydestä puhuttaessa voikin tarkemmin sanoa, että Lean-ajattelussa pyritään saavuttamaan täydellisyys jatkuvilla parannuksilla, jotka poistavat jätettä. Parannukset tapahtuvat yleensä kaikakun eli lyhyessä ajassa tapahtuvan radikaalien parannusten, tai kaizenin, eli pienempien, jatkuvien parannusten, kautta.

2.4 Lean-periaatteet ohjelmistotuotannossa

Kuten aiemmin on mainittu, Leanin juuret ovat autoteollisuudessa. TPS:n periaatteiden esittelyssä korostettiin, että tuotannon tasaisuus ja työtehtävien standardoitavuus ovat oleellisia osia tehokkaan arvovirran aikaan saamiseksi. Autoteollisuudesta poiketen, ohjelmistokehitys on luonteeltaan hyvin erilaista. Työ on usein uuden tuotteen kehittämistä ja jo tuotannossa olevien järjestelmien ylläpitokaan harvoin on standardoitavissa ja toistavaa. Jatkuva muutos on pikemminkin sääntö kuin poikkeus [HED93].

Mary ja Tom Poppendieck [Pop06] vertaavatkin ohjelmistokehitystä uuden autokonseptin kehittämiseen. Tätä varten Toyota on kehittänyt järjestelmän nimeltä Toyota Product Development System (Lean-tuotekehitys). Vaikka työtehtävät saattavat luonteeltaan poiketa vahvasti Lean-tuotannosta, Clark ja Fujimoto [ClF91] löysivät huomattavia samankaltaisuuksien näiden välillä. Vertailun yhteenveto on taulukossa 3.

Lean-tuotanto	**Lean-tuotekehitys**
Jatkuvia muutoksia tuotannon konfiguraatioon	Jatkuvia muutoksia tuotteisiin
Lyhyt tuotannon läpimenoaika	Lyhyt kehitysaika
Vähän keskeneräisten töiden varastointia tuotantovaiheiden välillä	Vähän tiedon varastointia kehitysvaiheiden välillä
Jatkuvaa pienten komponenttierien välitystä tuotantovaiheiden välillä	Jatkuvaa pienten esitietojen välitystä kehitysvaiheiden välillä
Pienemmät varastot vaativat joustavia resursseja ja suurempaa tiedonvälitystä tuotantovaiheiden välillä	Pienempi kehitysaika vaatii joustavia resursseja sekä tiedonvälitystä kehitysvaiheiden välillä
Joustavuus muuttaa tuotantomääriä, valmistettavia tuotteita ja tuotteiden määrityksiä	Joustavuus muuttaa tuotteiden määrityksiä, aikataulua ja tuotantokustannustavoitteita
Laajat tehtävämääritykset, jotka antavat tuotantotyöntekijöille enemmän toiminnanvapautta, parantavat tuottavuutta	Laajat tehtävämääritykset, jotka antavat kehittäjille enemmän toiminnanvapautta, parantavat tuottavuutta
Keskittyy nopeaa ongelmanratkontaan sekä prosessien jatkuvaan parantamiseen	Keskittyy jatkuvien, asteittaisten innovaatioiden sekä tuotteiden ja prosessien jatkuvaan parantamiseen
Laadun, toimitusajan ja tuottavuuden samanaikainen parantaminen	Laadun, kehitysajan sekä kehitystuottavuuden samanaikainen parantaminen

Taulukko 3: Yhteenveto Lean-tuotannon ja Lean-tuotekehityksen vertailusta

Poppendieckien mukaan, koska ohjelmistokehitys on verrattavissa tuotekehitykseen, ja Lean-tuotanto ja Lean-tuotekehitys ovat lähellä toisiaan, voidaan Leanin periaatteita hyödyntää myös ohjelmistokehityksessä [Pop06].

Poppendieckien ohjeet Leanin soveltamiseen ohjelmistokehityksessä pohjautuvat vahvasti Womack ja Jonesin [WoJ98] Lean-ajatteluun ja on ottanut inspiraatiota myös TPS:stä. Pyrkimys tasaiseen arvovirran soljuntaan on sama, mutta he määrittelivät Lean-ohjelmistokehitykselle 7 erilaista periaatetta seuraavasti:

- Poista hukka (Eliminate Waste)
- Tee laadusta sisäsyntyistä (Build Quality In)
- Tehosta oppimista (Create Knowledge)
- Lykkää päätöksentekoa (Defer Commitment)
- Toimita nopeasti (Deliver Fast)
- Kunnioita ihmisiä (Respect People)
- Optimoi kokonaisuus (Optimize The Whole)

Poista hukka

Kuten Lean-ajattelussa ja TPS:ssä, hukan poistaminen arvovirrasta on olennainen osa tehokasta Lean-ohjelmistokehitystä. Menetelmät ja vaatimus kaikkien tuotantoprosessiin osallistuvien valppaudesta ovat samat. Poikkeuksena Poppendieckit ovat määritelleet tarkkailtavat hukat paremmin ohjelmistokehitykseen soveltuviksi. Nämä esiteltiin aliluvussa 2.1.

Tee laadusta sisäsyntyistä

Poppendieckien mukaan laadun tulisi rakentaa koodiin alusta alkaen, ei vasta testivaiheessa. Tyypillinen tapa toimia virheiden kohdalla on tehdä asiasta virheraportti ja laittaa se virheraporttien seurantajärjestelmään. Aikanaan virheraportit tarkastetaan ja virheet korjataan. Kaikki virheet ovat kuitenkin keskeneräistä työtä eli hukkaa. Näin tavoitteena tulisikin olla jatkuvan integroinnin sekä yksikkö- ja automaattotestauksen avulla poistaa seurantajärjestelmät. Lisäksi tulisi harjoittaa jidokaa, eli pysähtyä ja ratkaista ongelmat välittömästi.

Vaikka ongelmat korjattaisiin välittömästi ja vikojen määrä pysyisi pienenä, Poppendieckiet kuitenkin suosittelevat erillistä hyväksymistestaus tai verifointivaihetta, Mikäli nämä vaiheet jatkuvasti aiheuttavat testaus ja korjaus – syklejä, on kehitysprosessissa jotain vikaa.

He myös mainitsevat, että helpoin tapa parantaa laatua ja poistaa hukkaa on kirjoittaa mahdollisimman vähän koodia. Mitä yksinkertaisempaa ja selkeämpää koodi on, sitä vähemmän siinä vikoja ja sen helpompaa sen päälle on jatkossa rakentaa. Heidän mukaansa tulisikin löytää se osa 20 % koodista, joka tuottaa 80 % arvosta, ja toteuttaa ensin se.

Tehosta oppimista

Perinteisessä vesiputousmallissa oletetaan, että kaikkia tarvittava tietoa rakennettavan järjestelmän suunnittelemiseksi on jo olemassa. Näin ollen järjestelmä voidaan määritellä, ja suunnitella, jo ennen toteutuksen aloitusta. Käytännössä suunnitelma kuitenkin elää toteutuksen aikana, sillä kaikkia yksityiskohtia ja tuotantoprosesseihin liittyvää monimutkaisuutta on käytännössä mahdotonta ennustaa etukäteen. Poppendieckien mukaan onkin erittäin tärkeää, että kehitysprosessit sallivat suunnitelmien muuttamisen, niin kehittäjien kuin asiakkaiden palautteen perusteella.

Muutokset yleensä perustuvat siihen, että kehittäjä tai asiakas on oppinut jotain uutta liittyen kehitettävään järjestelmään. Tämän kaltaisen tiedon voidaan antaa syntyä itsestään tai tätä tietoa voidaan aktiivisesti luoda. Poppendieckien mukaan monia menestyviä yrityksiä yhdistääkin yksi piirre: ne luovat tietoa järjestelmällisillä kokeiluilla ja siirtävät luodun tiedon koko organisaation käyttöön. Tietoa tulisikin jatkuvasti tallentaa ja tehdä eksplisiittiseksi, jotta sitä voidaan jakaa, eikä se elä vain muutaman ”gurun” pään sisällä.

Lykkää päätöksentekoa

Yksinkertaistettuna tämän periaatteen mukaan päätökset tulisi tehdä mahdollisimman myöhäisessä vaiheessa, sillä mitä myöhemmin päätös tehdään, sitä enemmän tietoa on saatavilla. Poppendieckien mukaan varhaisella päätöksenteolla pyritään vähentämään riskejä poistamalla tuntemattomia tekijöitä. Päätöksenteossa ensisijaisena periaatteena tulisi kuitenkin olla sellaisten valintojen tekeminen, joita voidaan muuttaa ja muokata helposti myöhemmin. Jotkin päätökset ovat kuitenkin peruuttamattomia, ja näiden tekemistä pitäisi lykätä niin myöhäiseen vaiheeseen kuin vain mahdollista.

Heidän mukaansa ennen aikaista päätöksentekoa toimivampi tapa on kokeilla monilla vaihtoehdoilla rinnakkain, näin mahdollistaen kriittisten valintojen tekemisen lykkäämisen viimeiseen mahdolliseen hetkeen. Tätä kutsutaan myös Set-Based Designiksi.

Toimita nopeasti

Mitä nopeammin asiakkaalle voidaan toimittaa tuotteita, sitä nopeammin niistä saadaan palautetta ja halutut muutokset voidaan toteuttaa seuraavissa tuotteissa. Mikäli toimitus tiedetään tapahtuvan nopeasti, voidaan myös lykätä päätöstä tuotannon aloittamisesta myöhemmäksi. Tällä varmistetaan myös se, että asiakkaalle tuotetaan juuri sitä, mitä hän tarvitsee nyt, eikä sitä mitä hän tarvitsi 6 kuukautta sitten. Näin toimilla saadaan

myös vähennettyä käynnissä olevan, eli keskeneräisen, työn määrää. Testaamaton ja integroimaton työ saattaa sisältää vikoja, ja näin kasvattaa tuotteeseen kohdistuvia riskejä.

Kunnioita ihmisiä

Työntekijöille tulisi antaa valtaa ja vastuuta omasta työstään. Ylhäältä alaspäin tapahtuvan käskyttämisen sijaan heille tulisi asettaa realistisia tavoitteita sekä antaa resurssit tavoitteiden saavuttamiseen. Työntekijöiden tulisi sallia käyttää omaa osaamistaan ja järkeään siihen, kuinka hänelle asetetut tavoitteet parhaiten saavutetaan.

Tätä tukee oppiva ja itsetarkastelua harjoittava organisaatio. Kuten TPS:ssä, organisaation tulisi myös panostaa omien työtekijöidensä osaamisen kasvattamiseen ja ylläpitoon, osaamisen ulkopuolelta hankkimisen sijasta.

Optimoi kokonaisuus

Tästä esimerkkinä Poppendieckit antavat tilanteen, jossa testaajilla on liikaa töitä. Tämä johtaa siihen, että valmiiden koodien testien valmistuminen pitkittyy, kehittäjät eivät saa palautetta riittävän ajoissa, koodiin syntyy virheitä ja testaajille syntyy entistä enemmän työtä. Näin ollen projekti viivästyy ja tuotteen laatu heikkenee.

Tästä syystä kun toimintaa tehostetaan, pitäisi tehostustoimissa ottaa huomioon koko tuotantoprosessi yksittäisten toimintojen sijasta. Mikäli näin ei toimita, saattaa yhden toiminnon tehostuminen tapahtua muiden toimintojen kustannuksella, näin laskien kokonaistehokkuutta.

2.5 *Aiemmat tutkimukset hukan havaitsemisesta*

Tutkielman teoriaan tutustumisen yhteydessä tarkasteltiin myös aiheeseen liittyviä aiempia tutkimuksia. Näitä etsittiin epäformaalein menetelmin, internetportaalien IEEE Xploren [IEE15], SpringerLinkin [Spr15], ScienceDirectin [Sci15] sekä ACM Digital Libraryn [ACM15] kautta hakusanoilla *Lean waste case study*, *lean waste*, *lean case study.*

Näistä lähteistä löytyi tapaustutkimuksia, joissa esitellään Leanin implementointia niin ohjelmistokehityksessä [RPK14], [MiJ12] kuin nougatin valmistuksessa [TSR13]. Myös ehdotelmia kehikoiksi Leanin hallintaan [PiM14], Lean-prosessien parantamiseen [PeM10] sekä hukan määritelmäksi [PoC14] ohjelmistokehityksen kontekstissa löytyy, mutta raportteja, joissa hukkaa havainnoidaan ja poistetaan, löytyi tämän etsinnän yh-

teydessä vain muutamia.

Tarkasteluhetkellä tutkielman tavoitteita vastaavia tapaustutkimuksia löytyi kolme: Ikonen et al.[IPO10], Mujtaba et al.[MDP10] sekä Petersen [Pet12]. Tutkimuksessaan Ikonen et al. pyrkivät tunnistamaan Poppendieckien [Pop06] määrittelemiä 7 hukkaa Kanban-prosessia käyttäneestä ohjelmistokehitysprojektista ja tutkivat, vaikuttaako hukka projektin onnistumiseen. Hukkaa havainnoitiin haastattelemalla kehittäjiä. Näiden kautta jokaisen luokan mukaista hukkaa havaittiin, mutta näiden ei koettu vaikuttaneen projektin onnistumiseen. Hukan määrää tai sen vaikutusta projektin tehokkuuteen ei mitattu

Mujtaba et al. [MDP10]pyrkivät tunnistamaan hukkaa analysoimalla Ericcson AB:lla tapahtuvan ohjelmistojen kustomointiprosessin arvovirtakarttaa. He aluksi tutkivat kohdeprosessiin liittyvää dokumentaatiota ja haastattelivat prosessin työntekijöitä. Näiden tietojen perusteella he loivat prosessille arvovirtakartan. Tämän jälkeen he keräsivät yhtiön järjestelmistä prosessiin liittyvää historiallista dataa ja laskivat läpimenoaikoja prosessin eri osille. Lopuksi he haastattelivat työntekijöitä uudelleen varmentaakseen arvovirtakartan oikeellisuutta. Läpimenoaikojen analyysi paljasti vain odottelusta johtuvaa hukkaa. Tämä pystyttiin kuitenkin mittaamaan ja sen todettiin vievän 38 % koko prosessiin käytetystä ajasta. Työntekijöiden haastattelut myös paljastivat ylimääräisestä prosessoinnista sekä liikkeestä johtuvaa hukkaa. Näistä hukista ei annettu määrällistä arviota.

Petersen [Pet12] puolestaan tutki hukkaa ohjelmistojen ylläpidon yhteydessä. Hän tutki erästä Ericsson AB:n tuotetta ja siihen liittyvien ylläpitopyyntöjen käsittelyä. Pyynnöt oli lajiteltu niiden vakavuuden sekä teknisen haastavuuden mukaan kolmeen kategoriaan alkaen vaikeimmasta ja päättyen helpoimpaan: A, B ja C.

Petersen tutki ylläpitoprosessin tehokkuutta ja vaikuttavuutta neljän indikaattorin avulla: ylläpitopyyntöjen määrä per aikayksikkö (esimerkiksi viikko), virtauksen visualisointi kumulatiivisen arvovirtakartan avulla, läpivirtausaika sekä työn alla olevien ylläpitopyyntöjen määrä tietyllä ajanhetkellä

Petersen analysoi näitä indikaattoreita yhdessä ja huomasi, että pyyntöjen loppuunsaattamisen odottaminen oli suurin pullonkaula prosessissa ja tämän poistoa tulisi priorisoida. Hän huomasi myös, että 50 % koko prosessiin käytetystä ajasta oli odotusta.

3 Tapaustutkimuksen tutkimuskohteen esittely

Tapaustutkimuksen kohde on keskisuuren valtiollisen toimijan sisäisen IT-yksikön ohjelmistokehitystoimiston ja sen sidosryhmien yhteistyössä toteuttama ohjelmistokehitysprojekti. Toimijan pääasiallinen tehtävä liittyy talouden ja rahapolitiikan valvontaan sekä ylläpitoon.

3.1 *Kohdeorganisaatio*

Tätä tehtävää varten toimija kokonaisuudessaan työllistää 400 henkilöä, joista suuri osa on taloustieteiden asiantuntijoita. Kuten monissa asiantuntijatehtävissä tänä päivänä, näiden asiantuntijoiden työkalut ovat pääasiassa digitaalisia. Osa näistä on sellaisinaan käyttöönotettuja valmisohjelmistoja, mutta suurinta osaa muokataan vastaamaan organisaation erikoistuneita tarpeita.

Kaikkia asiantuntijoiden tarpeita ei kuitenkaan ole pystytty kattamaan muokkaamalla valmiista ohjelmistoja. Ohjelmistojen tilaaminen ulkopuolista toimittajilta ei myöskään aina ole toiminut toivotulla tavalla, sillä tarvittavien ohjelmistojen kehittäminen vaatii syvää asiantuntijuutta Suomessa uniikilla liiketoiminta-alueella.

Tämän takia toimijalla on oma sisäinen IT-yksikön, jonka tehtävänä on tukea asiantuntijoiden työskentelyä kokonaisvaltaisesti liittyen IT-palveluihin, aina salasanojen vaihdosta uusien ekonomistiohjelmien tuottamiseen. Yksikössä työskentelee noin 60 henkilöä jaettuna kolmeen toimistoon: infrastruktuuri(infra)-, palvelu- sekä ohjelmistokehitystoimisto. Infratoimisto vastaa nimensä mukaisesti IT:hen liittyvästä infrastruktuurista, käsittäen niin sisäverkkojen ja konesalien fyysisen ylläpidon kuin tuotannossa olevien ohjelmistojärjestelmien toiminnan valvonnan. Palvelutoimiston tehtäviin kuuluu muun muassa fyysisten työasemien ylläpito, valmisohjelmistojen lisenssien hallinta sekä IT-tuen antaminen käyttäjille.

Tämän tutkimuksen tarkastelun kohteena oleva ohjelmistokehitystoimisto työllistää 23 henkilöä, jonka lisäksi toimisto käyttää laajasti ulkopuolisia konsultteja, noin 16 työvuoden edestä vuosittain. Toimiston tehtävänä on kehittää ja ylläpitää toimijan koko organisaation tarvitsemia ohjelmistoja.

Osa ylläpidosta tapahtuu projektien ulkopuolella niin sanottuna linjatyönä. Järjestelmille on määrätty yksi tai muutama vastuuhenkilöt, jotka vastaavat virheiden ja päivitysten tekemisestä. Suuri osa uusien ohjelmistojen kehitystyöstä ja noin 46 % kaikesta järjes-

telmätoimiston työstä tehdään projekteissa. Projektien koko vaihtelee yhden kehittäjän kuukauden työpanoksesta kymmenen hengen tiimin usean vuoden rupeamaan.

Projektit syntyvät asiakkaana toimiva osasto ilmaistessa tarpeen tietylle työkalulle. Tarve kirjataan hankesalkkuun. Salkun sisältöä käsitellään IT-yksikön ja asiakasosaston välisissä tapaamisissa muutamia kertoja vuodessa, ja tällöin tarpeen toteuttamisesta voidaan sopia. Mikäli se nähdään tarpeelliseksi, tarpeen toteutus voidaan suorittaa projektina, ja projekti käynnistämisestä päätetään sidosryhmien yhteisellä päätöksellä. Tyypillisesti projekteihin osallistuu asiantuntijoita niin IT kuin asiakasosastoiltakin.

3.2 *Tämän hetkinen kehitysmalli organisaatiossa*

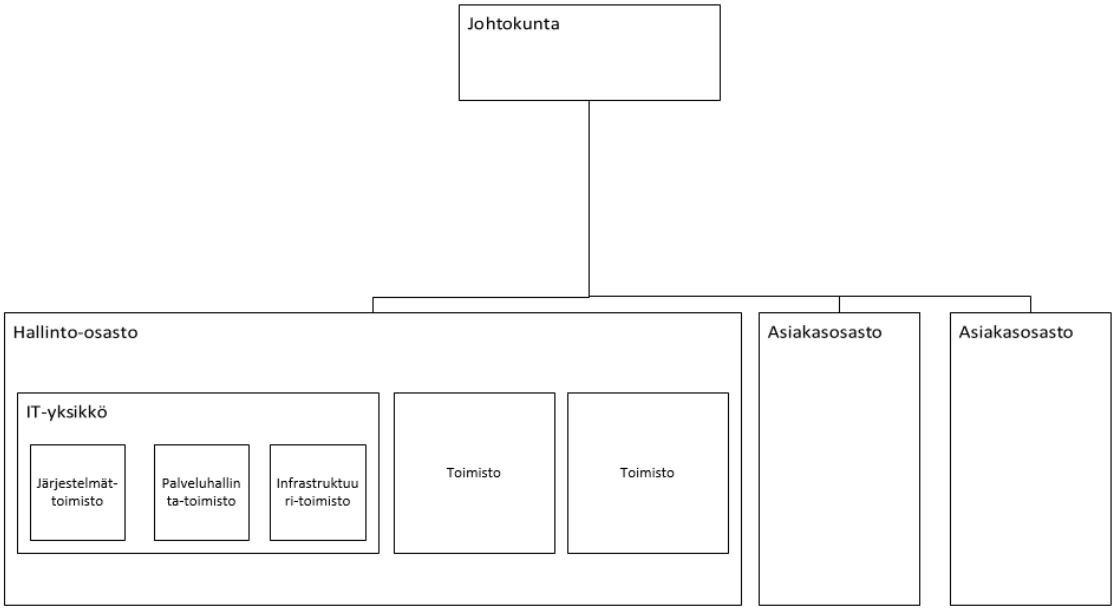
IT-yksikön projektien tehokkuutta on pyritty parantamaan ottamalla käyttöön ketteriä kehitysmenetelmiä. Erilaisia tekniikoita on kokeiltu kehittäjien ja projektipäälliköiden toimesta. Tällä hetkellä projekteissa hyödynnetään ainakin seuraavia tekniikoita:

- **Kanban**
- **Käyttäjäkertomukset/käyttötapaukset**
- **Scrum**
- **Scrum Review**
- **Jälkipuinti**
- **Continous Integration**
- **Automaattitestaus**
- **Code review**

Kuten ketteryyteen kuuluu, mitään käytäntöjä ei pakoteta suoraan projekteille vaan projektin tiimillä on vapaus soveltaa haluamiaan käytäntöjä parhaaksi katsomallaan tavalla. Kehittäjät tulevat talon sisältä tai tarpeen mukaan palkataan talon ulkopuolisia konsultteja. Tiimi pyritään rakentamaan ketterien periaatteiden mukaisesti niin, että tiimistä löytyy kaikki tarvittava osaaminen projektin läpiviemiseksi. Asiakasosastot vastaavat yleensä projektien vaatimusten tuottamisesta sekä ohjelmistojen testaamisesta.

Projektitiimit saavat myös tukea muilta toimistoilta palveluiden muodossa. Esimerkiksi kehitys-, testaus- ja tuotantoympäristöjen ylläpito tapahtuu pääosin infratoimistossa. Projektista syntyvän tuotteen omistaja määrätään asiakasosastolta, ja perinteisestä ketterästä mallista poiketen projektilla on yleensä projektipäällikkö. Kuvassa 2 on esitelty

toimijan organisaatiomalli.

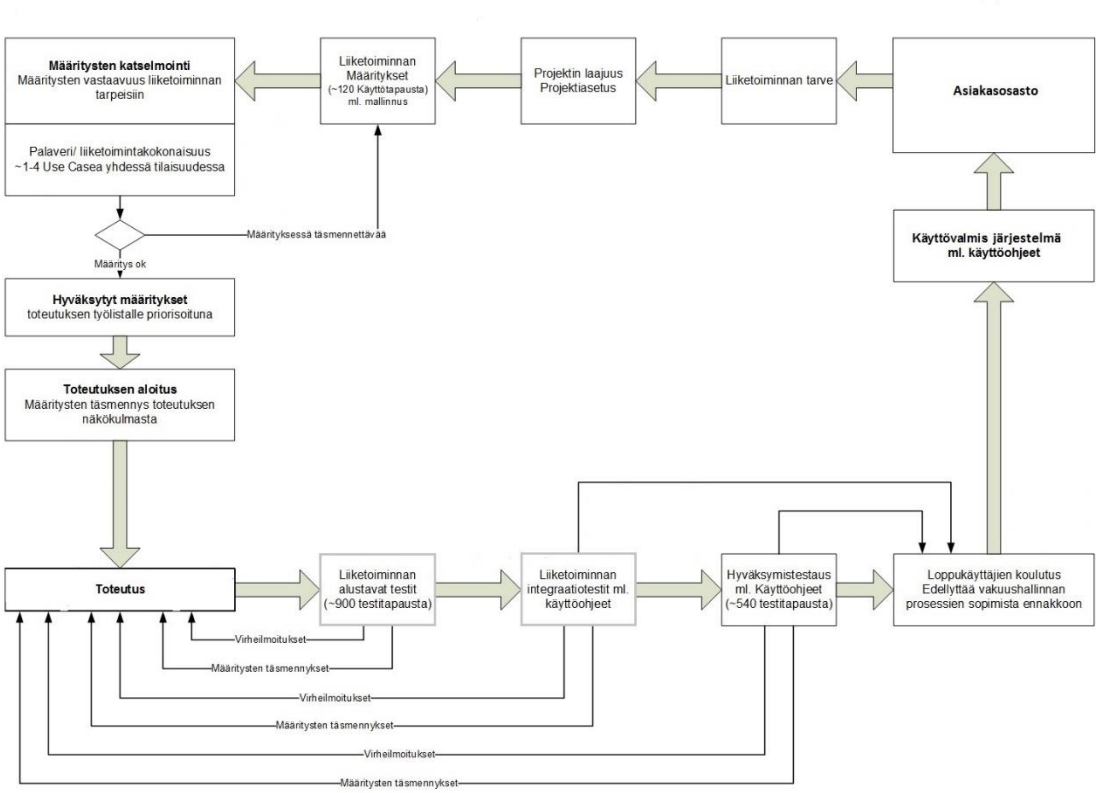


Kuva 2: Toimijan organisaatiomalli

Koska toimijan lähes koko organisaatio työskentelee fyysisesti lähekkäin olevissa rakennuksissa, kehitystiimit sekä asiakasosasto työskentelevät lähellä toisiaan. Tästä syystä asiakaskommunikaatio on pääsääntöisesti tehokasta, ja tässä tutkielmassa tarkastelussa projektissa kommunikointia parannettiin entisestään. Parannusta käsitellään tarkemmin tutkimustulosten yhteydessä luvussa 5.

3.3 *Kohdeprojekti*

Projekti aloitettiin joulukuussa 2011 ja sen tuottaman järjestelmän ensimmäinen versio valmistui joulukuussa 2014. Itse projekti päättyi maaliskuussa 2015. Projekti oli yksi niin IT-yksikön kuin asiakasosaston suurimpia ja kriittinen organisaation toiminnalle. Siihen osallistui vuosien saatossa 10 organisaation omaa kehittäjää ja 10 konsulttia. Asiakasosastolta vaatimusmäärittelyyn ja testaukseen on osallistunut 15 henkilöä. Ajanhetkestä toiseen kehitystiimin koko on vaihdellut 4 ja 11 välillä, kun taas projektitiimin koko on vaihdellut 15 ja 20 välillä. Sen tarkoituksena oli automatisoida toimijan asiakkaiden käyttämiä prosesseja, joilla hallinnoidaan merkittäviä määriä varallisuutta. Nämä prosessit ovat toimijalle kriittisiä, ja siksi ohjelmistokehitysprojekti oli myös erittäin tärkeä. Lopullinen järjestelmä sisältää noin 200 000 riviä koodia. Kuvassa 3 on yleiskuva projektin vaiheista



Kuva 3: Yleiskuva kohdeprojektin vaiheista

Projektin ensimmäinen työvaihe oli liiketoiminnan tarpeiden kirjaaminen käyttötapaus-dokumentteihin. Käyttötapausten tarkoituksena on kuvailla askeleita tai tapahtumia, jotka tapahtuvat jotta järjestelmä pääsee tavoiteltuun lopputulokseen. Cockburn [Coc02] kuvailee käyttötapauksia proosallisiksi kuvauksiksi järjestelmän käyttäytymisestä kun se toimii yhdessä ulkopuolisen maailman kanssa.

Kuten kuvan 3 vasemmasta ylälaidasta nähdään, vaatimusmäärittelijät kirjoittivat käyttötapauksia iteratiivisesti ja niille oli oma hyväksymisprosessinsa. Kaikki käyttötapaukset pyrittiin kirjoittamaan valmiiksi ennen kuin yhtään käyttötapausta otettiin toteutukseen. Itse toteutusvaihe noudatti sovellettua Scrum-mallia. Kun käyttötapaukset olivat valmiit, niiden sisällöt purettiin tehtäviksi ja jaettiin sprintteihin.

Kokonaisuudessaan projektin käyttötapaukset kävivät läpi seuraavat vaiheet:

1. **Käyttötapauksen määritys ja mallinnus:** Asiakasosastolta osoitetut määrittelijät loivat asiakasosaston tarpeiden perusteella käyttötapauksia, jotka kirjattiin projektitiimin yhteiseen sähköiseen työtilaan.
2. **Käyttötapausten katselmointi ja hyväksyntä:** Valmiit käyttötapaukset katselmoitiin erillisissä katselmointikokouksissa, johon osallistuivat vaatimusmäärittelijät, projektipäällikkö sekä tarvittaessa muita liiketoiminnan asiantuntijoita. Mi-

käli ongelmia ei havaittu, projektipäällikkö kirjasi hyväksytyn käyttötapauksen projektin työlistalle (product backlog) priorisoituina.

3. **Käyttötapauksen purku tehtäväksi:** Projektipäällikkö tai kehittäjä analysoi käyttötapausta ja kirjasi tämän pohjalta tehtäviä tiimin sähköiselle Scrum-taululle.
4. **Käyttötapauksen toteutus:** Tähän kuului niin itse logiikan kuin testien ohjelmointi. Koska projektissa käytettiin jatkuvaa integrointia, toteutuksen piti myös läpäistä järjestelmän kriittiset osat kattava automaattinen testiajo ennen kuin se hyväksyttiin toteutetuksi.
5. **Testaus:** Toteutusvaiheen automaattitestien jälkeen asiakasosastolta määrätyt testaajat verifioivat käyttötapauksen toteutuksen manuaalisilla hyväksymistesteillä, jotka asiakasosaston määrittelijät olivat aiemmin laatineet. Testit oli laadittu niin, että keskimäärin yhden käyttötapauksen testaaminen onnistui yhdessä työpäivässä tai vähemmässä
6. **Käyttöönotto:** Toteutetut käyttötapaukset vietiin ensin kehittäjien toimesta testiympäristöön, jossa liiketoiminnan edustajat ajoivat käyttötapauksiin liittyvät testit. Kun käyttötapaukset oli hyväksymistestattu, ne vietiin yhtenä kokonaisuutena tuotantoon.

Projektin vaiheita toteutettiin osittain vain kehittäjien toimesta, osittain vain liiketoiminnan edustajien toimesta ja osittain yhteistyössä. Mikään vaihe ei ollut puhtaasti sidottu tiettyyn toimistoon.

Projektissa käytettiin iteratiivista kehitystapaa. Näin ollen tietty käyttötapaus saattoi 5. vaiheen jälkeen joko palata virheiden takia vaiheeseen 4., tai jos määrityksissä huomattiin muutostarpeita, palata vaiheeseen 1.

Käyttötapausdokumentit kirjoitettiin Microsoftin Word-ohjelmalla ja tallennettiin Microsoftin Sharepoint-alustalle rakennettuun sähköiseen työtilaan. Kumpikin näistä mahdollistaa versionhallinnan, jossa päivämäärän ja kellonajan sisältävällä aikaleimalla merkitään hetki, jolloin muutokset dokumenttiin on tallennettu.

Käyttötapauksiin liittyviä työtehtäviä ja niiden vaiheita käsiteltiin Microsoftin Team Foundation Server-alustan päälle tehdyssä sähköisessä työtilassa. Työtila koostui Scrum-taulusta sekä erillisistä näkymistä, joissa projektin henkilöstö pystyi muokkaamaan ja tarkastelemaan yksittäisiä työtehtäviä edustavia kortteja. Näillä sähköisillä

Scrum-kortteilla oli tilajärjestelmä, joka näytti sen tämänhetkisen työvaiheen, sekä työvaihehistorian. Esimerkiksi kun projektipäällikkö loi kortin, oli se tilassa ”Created”. Kun se otettiin toteutukseen, sen tilan muutettiin tilaan ”Commited” ja kun työ oli valmis, meni se tilaan ”Done”. Jokaiseen työvaiheeseen liittyi myös päivämäärän sisältävä aikaleima, jossa oli myös muutokseen tehneen henkilön tunnus. Vaikkakin järjestelmä mahdollisti jokaisen työvaiheen leimaamisen aloitus-, keskeytys ja lopetusajalla, järjestelmää hyödynnettiin tapausprojektissa vaillinaisesti, eikä kaikkia työtehtäviä ja niiden vaiheita merkitty.

3.4 *Tutkimuskohteen oletettu hukka*

Kohdeprojektissa oletettiin ilmenevän hukkaa monessa muodossa, ja erityisesti virheiden sekä viivytysten muodossa. Nämä olettamukset muodostuivat siitä, että tutkimuskohteena olevan projektin ollessa käynnissä, projektiin osallistuneiden henkilöiden välisissä epävirallisissa keskusteluissa nousi jatkuvasti esiin kaksi asiaa: toteutuksen virheiden määrä sekä käyttötapauksiin liittyvät ongelmat. Myös tutkielman laatijan henkilökohtaiset havainnot viittasivat siihen, että nämä ongelmat aiheuttavat turhaa työtä. Projektin ollessa käynnissä työtehtävät myös keskeytyivät usein, koska erityisesti kehittäjät joutuivat odottamaan työn jatkamiseen tarvittavaa tietoa.

Nämä laatijan henkilökohtaiset havainnot vahvistuivat projektin puitteissa järjestetyssä laadunvarmistustapaamisessa, jossa koko projektihenkilöstö pohti projektin ongelmakohtia. Tapaamisessa nousi esille vastaavia huomioita. Tapaaminen esitellään aliluvuissa 4.5.3. sekä sen tulokset aliluvussa 5.2.

4 Empiirinen tutkimus

Tässä tutkielmassa suoritettu tapaustutkimus noudattaa Runesonin ja Höstin [RuH09] ehdottamaa ohjeistusta tapaustutkimuksen suorittamisesta.

4.1 *Tutkimusmenetelmät*

Robsonin [Rob02], Yin [Yin02] ja Benbasat et al. [BGM87] mukaan tapaustutkimus on tutkimusmenetelmä, jonka tarkoituksena on tutkia tiettyä ilmiötä sen omassa kontekstissa. Robson näkee sen olevan tutkimusstrategia ja painottaa useiden lähteiden käyttämisen tärkeyttä. Yin luonnehtii sitä tiedusteluksi ja sanoo että raja tutkittavan ilmiön ja kontekstin välillä on häilyvä. Benbasat et al. vuorostaan sanoo että tapaustutkimus on tiedon keräämistä muutamasta toimijasta (kuten ihmiset, ryhmät, organisaatiot) ja sen erityispiirteenä on kokeellisen kontrollin puute. Tapaustutkimus saattaa sisältää myös elementtejä muista tutkimusmenetelmistä kuten kyselytutkimuksesta tai kokeellisesta tutkimuksesta.

Runeson ja Höstin [RuH09] mukaan tapaustutkimukseen kuuluu viisi pääaskelta:

1. Suunnittelu: tavoitteet määritellään ja tapaustutkimus suunnitellaan
2. Tiedon keräyksen valmistelu: tiedon keräyksen menetelmät määritellään
3. Tiedon keräys: tieto kerätään tutkittavista tapauksista
4. Analysointi: kerätty tieto analysoidaan
5. Raportointi

Kyselytutkimukset voivat näkökulmasta riippuen palvella eri tarkoituksia. Robson [Rob02] luokittelee nämä neljään ryhmään:

- Tutkivat tutkimukset pyrkivät löytämään uusia näkökulmia asioihin ja luomaan ideoita uutta tutkimusta varten.
- Kuvaavissa tapaustutkimuksissa tarkoituksena on esitellä tilannetta tai ilmiötä.
- Selittävien tutkimusten päämääränä on löytää selitys ilmiölle tai ongelmalle, mahdollisesti kausaalisten suhteiden kautta.
- Parantavat tapaustutkimukset pyrkivät parantamaan tutkittuun ilmiöön liittyvää puolta.

Tapaustutkimusta varten kerätty tieto voi olla kvantitatiivista tai kvalitatiivista. Kvalita-

tiivisen tiedon käyttö on yleisempää, sillä se tarjoaa laajemman ja monipuolisemman näkökulman tutkittavaan ilmiöön. Kvalitatiivisen ja kvantitatiivisen tiedon yhdistelmä tarjoaa kuitenkin usein paremman ymmärryksen tutkittavasta ilmiöstä. Tämän takia tiedon triangulointi eli tiedon kerääminen useasta eri lähteestä on oleellista tutkimuksen edustavuuden ja validiuden parantamiseksi, erityisesti kvalitatiivisen tiedon kohdalla. Triangulaatioita voidaan tehdä neljällä tavalla:

- Tietolähteiden triangulointinissa tietoa kerätään joko useista lähteistä tai tietoa kerätään eri aikoina.
- Tarkkailijoiden triangulointinissa käytetään useampaa kuin yhtä tarkkailijaa.
- Menetelmällisessä triangulointinissa yhdistetään eri menetelmiä tiedon keräämisessä, esimerkiksi kvantitatiivista ja kvalitatiivista.
- Teorian triangluoinnissa käytetään useita teorioita tai näkökulmia.

Shull et al. [SSS08] meta-analyysin mukaan ohjelmistokehitykseen liittyvissä tutkimuksissa lopputulokseen vaikuttavaa monta tekijää. Tapaustutkimus keskittyvät tutkimaan ilmiöitä niiden kontekstissa, erityisesti silloin kun raja ilmiön ja kontekstin välillä on häilyvä. Tapaustutkimukset ovat myös joustavia ja pystyvät tehokkaasti tutkimaan dynaamisia ja monimutkaisia ilmiöitä, joten ne sopivat ohjelmistokehityksen tutkimukseen hyvin.

Kvantitatiivinen tutkimus

Kvantitatiivisen tutkimuksen tarkoituksena on kerätä, analysoida ja esittää tietoa numeerisessa muodossa tutkittavasta ilmiöstä [Giv08]. Pyrkimyksenä on kerätä riittävästi havaintoja, jotta niistä voidaan tilastollisella luotettavuudella yleistää koskemaan koko mitattavaa perusjoukkoa.

Kerättyä tietoa analysoidaan perinteisesti luomalla kuvaavia tilastoja, analysoimalla korrelaatioita, kehittämällä ennustavia malleja ja testaamalla hypoteeseja [RuH09]. Kuvaavia tilastoja kuten keskiarvoja, poikkeamia, histogrammeja ja parvikuvioita käyttämällä pyritään ymmärtämään kerättyä tietoa paremmin. Korrelaatioiden ja ennustavien mallien kehittämisen avulla pyritään löytämään yhteyksiä eri vaiheissa mitattujen arvojen välillä. Hypoteesien testauksella taas pyritään selvittämään millaisia vaikutuksia yksi tai useampi riippumaton muuttuja aiheuttaa yhteen tai useampaan riippuvaan muuttujaan.

Givenin [Giv08] mukaan kvantitatiivinen tutkimus nähdään usein kvalitatiivisen tutkimuksen vastakohta. Hänen mukaansa tämä vastakkainasettelu on kuitenkin harhaanjohtava, koska kvantitatiiviset tutkijat ovat kiinnostuneita ilmiöiden kvalitatiivisista ominaisuuksista ja kvalitatiiviset tutkijat eivät voi koskaan täysin välttää käyttämästä kvantitatiivisia menetelmiä. Kun he käyttävät adverbeja kuten *usein, joskus,* tai *ei koskaan*, he kvantifoivat tietoa.

Kvalitatiivinen tutkimus

Kvalitatiivinen tutkimus on suunniteltu tutkimaan kuinka ihmiset näkevät ja kokevat ympäröivän maailmaansa ja siihen liittyviä monimutkaisuuksia [Giv08]. Toisin kuin kvantitatiivisessa tutkimuksessa, kvalitatiivisessa tutkimuksessa kerätty tieto esiintyy sanoina ja kuvina, ei numeroina. Yleisiä tapoja tiedon keräämiseen ovat osallistuva tarkkailu, haastattelut ja kyselyt.

Osallistuvalla tarkkailulla tarkoitetaan tutkimusta, johon kuuluu sosiaalinen interaktio tarkkailijan ja tarkkailtavan välillä tarkkailtavan ympäristössä, jonka aikana tietoa systemaattisesti ja huomaamattomasti kerätään [Sea99]. Tarkkailija on siis avoimesti läsnä tarkkailtavien ympäristössä, ja nimestä huolimatta voi osallistua tai olla osallistumatta tarkkailtavaan käyttäytymiseen.

Robinsonin [Rob02] mukaan tarkkailun kautta voidaan saada syvää ymmärrystä tutkittavasta ilmiöstä, varsinkin jos ilmiöstä on olemassa niin sanottu ”virallinen” ja ”epävirallinen” näkemys. Seaman [Sea99] kuitenkin huomauttaa, että tarkkailun hyödyllisyyttä rajoittaa se, että suuri osa ohjelmistokehitykseen liittyvästä toiminnasta tapahtuu pään sisällä.

Seamanin [Sea99] haastatteluita käytetään esimerkiksi historiallisen tiedon, mielipiteiden tai vaikutelmien keräämiseen. Niitä voidaan käyttää myös yhdessä tarkkailun kanssa selventämään tarkkaillussa tehtyjä havaintoja.

Haastattelut voidaan toteuttaa joko strukturoidusti, strukturoimattomana tai puolistrukturoidusti. Strukturoidussa haastattelussa kaikki kysymykset esittää haastattelija ja haastateltava antaa vastaukset. Kysymykset, sekä järjestys missä ne kysytään, on etukäteen suunniteltu. Ne ovat myös tarkkaan rajattuja ja niillä on selkeä tavoite. Strukturoidun haastattelun tavoitteena on usein kuvailla tai selittää tutkittavaa ilmiötä.

Tämän vastakohta strukturoimattomassa haastattelussa haastattelun tavoitteet ja teemat on määritelty yleisemmällä tasolla, ja tarkoituksena on hankkia mahdollisimman paljon

tietoa aiheesta. Kysymykset ovat tyypillisesti avoimia ja haastattelu eteneekin enemmän keskustelun kaltaisena, jossa niin haastattelija kuin haastateltavakin voivat esittää kysymyksiä. Joissain tapauksissa haastelija ei edes esitä kysymyksiä vaan mainitsee vain haastattelun aiheen ja antaa haastateltavan kertoa asiasta. Haastattelun tavoitteena on usein tutkiva eli laajentaa ymmärrystä tutkittavasta asiasta.

Puolistrukturoitu haastattelu on nimensä mukaisesti näiden kahden välimalli, joka tyypillisesti koostuu sekä avoimista kysymyksistä että tarkkaan rajatuista kysymyksistä. Kuten strukturoidulla haastattelulla, sen tavoitteena on usein kuvailla tai selittää tutkittavaa ilmiötä.

Denscomben [Den10] mukaan tieteellisessä tutkimuksessa käytettävän kyselyn tarkoitus on kerätä tietoa analysointia varten ilman, että sillä pyritään vaikuttamaan mielipiteisiin tai jakamaan tietoa, toisin kuin esimerkiksi markkinoitaessa tuotteita. Kysely koostuu listasta kysymyksiä, jotka ovat identtisiä kaikille vastaajille ja jotka tarkkaan suunniteltuja. Denscomben [Den10] ja Giesen et al.[GMV12] mukaan kysymysten ja kyselyn yleensä tulisi noudattaa seuraavaa ohjeistusta:

- Kysy vain kysymyksiä joihin osallistujat voivat vastata
- Määrittele tarkkaan ketä ja mitä kysely koskee
- Kysy vain yksi kysymys kerrallaan
- Pyri esittämään kysymykset neutraalista näkökulmasta
- Älä tee perusteettomia olettamuksia kysymyksissä
- Käytä kielellisesti mahdollisimman selkeitä ilmaisuja, vältä teknistä sanastoa
- Tee kysymyksistä mahdollisimman lyhyitä ja suoraviivaisia
- Sisällytä vain ne kysymykset jotka ovat tutkimuksen kannalta välttämättömiä
- Vältä sanoja tai fraaseja jotka voidaan kokea loukkaaviksi

Kyselytutkimuksen hyviksi puoliksi voidaan lukea sen järjestämisen helppous ja halpuus, varsinkin sähköisessä muodossa. Kyselyt myös tuottavat suuria määriä tietoa standardoidussa muodossa. Lisäksi kerätty tieto on yleensä valmiiksi koodatussa muodossa, näin nopeuttaen tiedon käsittelyä, vaikkakin avoimia kysymyksiäkin voidaan kyselyissä käyttää.

Valmiin koodauksen haittana huonosti valitut vastausvaihtoehdot voidaan kokea rajoit-

tavina sekä turhauttavina, näin vähentäen vastaushalukkuutta. Vastausvaihtoehdot voivat myös heijastaa sitä, mitä tutkija haluaa ilmiössä nähdä kuin vastaajien kokemusta asiasta, näin tehden tutkimuksesta puolueellista. Tutkijalla on myös hyvin vähän työkaluja tarkastaa ovatko vastaajat vastanneet rehellisesti.

Seamanin [Sea99] mukaan kvalitatiivisen tiedon analysoinnilla pyritään tekemään päätelmiä, jotka muodostuvat selvistä todistusaineistojen ketjuista. Näin kenen tahansa pitäisi pystyä näkemään kuinka päätelmä syntyi. Kuten aiemmin on mainittu, tapaustutkimuksissa voidaan käyttää sekä kvantitatiivisia että kvalitatiivisia menetelmiä. Mikäli näiden eri menetelmien tuottamaa tietoa halutaan esimerkiksi tilastollisilla menetelmillä vertailla, kvalitatiivinen tieto voidaan koodata. Koodauksessa kuvan tai tekstin muodossa olevalle tiedolle annetaan numeraalisia arvoja. Koska kvalitatiivinen tieto on usein monipuolisempaa kuin kvantitatiivinen tieto, koodaus usein hävittää tietoa ja siksi sitä pitää tehdä harkiten ja huolella.

Analysointia tehdään kahdella eri tavalla: hypoteeseja luovilla tekniikoilla ja hypoteeseja varmistavilla tekniikoilla [Sea99]. Hypoteeseja luovilla tekniikoilla, kuten niin sanotulla ”jatkuvalla vertailulla” tai tapauksien ristivertailulla, pyritään luomaan toteamuksia tai väitöksiä tutkimalla muistiinpanoja. Jatkuvassa vertailussa muistiinpanoista poimitaan katkelmia, jotka sisältävät tutkijoita kiinnostavia teemoja tai ideoita, ja ne koodataan. Koodatut tekstit lajitellaan koodauksen mukaan ja näiden perusteella luodaan hypoteeseja. Seamanin [Sea99] mukaan ideaalitilanteessa tiedon keräystä ja koodausta tehdään useaan otteeseen, ja näin luotuja hypoteeseja vertaillaan aiemmin saatuihin. Tapauksien ristivertailussa kerättyä tietoa ryhmitellään esimerkiksi tiettyjen ominaisuuksien mukaan. Eri ryhmiä voidaan vertailla ryhmien sisäisten samankaltaisuuksien ja ryhmien välisten poikkeavuuksien löytämiseksi.

Seamin [Sea99] mukaan hypoteeseja varmistavilla tekniikoiden tarkoituksena on kasata todistusaineistoa, joka tukee luotua väitettä, ei todistaa onko väite tosi. Aiemmin esitellyt triangulaatiot ja tutkimuksen toistaminen ovat esimerkkejä hypoteesin varmistamisesta.

Robson [Rob02] lajittelee lähestymistavat tiedon analysointiin neljään kategoriaan perustuen niiden formaalisuuteen. Kategoriat on listattu alle alkaen vähiten formaalista ja päättyen formaalimpaan:

- Immersiiviset, jotka ovat rakenteellisesti kevyitä lähestymistapoja ja nojaavat vahvasti tutkijan näkemykseen sekä tulkintakykyyn.

- Editoivat, joissa tiedon koodaus muuttuu analysoinnin myötä
- Sapluunat, joissa tiedon koodaustavat on muodostettu etukäteen hypoteesien pohjalta
- Osittais-tilastollinen, jossa lasketaan esimerkiksi sanojen frekvenssejä

Runesonin ja Höstin [RuH09] mukaan editoivat ja sapluuna-tyyppiset lähestymistavat ovat osoittautuneet sopivimmiksi tapaustutkimuksia tehdessä.

4.2 *Tutkimussuunnitelma*

Rajallisten resurssien vuoksi tähän tutkielmaan on valikoitu vain yksi projekti, joka liiketoimintaloogisen sekä teknisen monipuolisuutensa ja kokonsa vuoksi koettiin olevan mahdollisimman edustava. Tutkimuksen kohteena olevasta projektista valitaan muutama mahdollisimman edustava käyttötapaus. Projektin sähköisistä järjestelmistä kerätään mahdollisimman paljon kvantitatiivista tietoa valittujen käyttötapausten eri työtehtäviin käytetystä työajasta. Kuten aliluvussa 4.1. esiteltiin, riittävällä kvantitatiivisella todistusaineistolla voidaan tutkittava ilmiö yleistää koskemaan koko perusjoukkoa. Tämän tutkielman tavoitteena onkin pystyä parantamaan tehokkuutta kaikissa IT-yksikön projekteissa, ei vain kohteena olevassa projektissa. Kun tiedot on kerätty, niistä muodostetaan jokaiselle käyttötapaukselle arvovirtakartta, jolla hukkaa pyritään havaitsemaan.

Kuten aliluvussa 4.1. mainittiin, kvalitatiivisella tiedolla voidaan saada syvempi ymmärrys tutkittavasta ilmiöstä. Tässä tutkielmassa kvantitatiivista tietoa tuetaan kvalitatiivisella tiedolla, jota kerätään haastattelemalla projektiin osallistuneita henkilöitä, tarkastelemalla projektiin liittyvän laadunvarmistustapaamisen muistiinpanoja sekä projektiin osallistuneen, tämän tutkielman laatijan, omilla havainnoilla. Haastattelut ovat luonteva tapa kerätä implisiittistä tietoa projektiin osallistuneilta henkilöiltä. Dokumenttien sisältöanalyysillä taas voidaan pienellä vaivalla saada merkittävää tietoa hukkien luonteesta.

Tutkimuksen yhteydessä suoritetaan myös kysely, jolla kartoitetaan kuinka paljon ja minkä tyyppisiä hukkia organisaation eri IT-projekteissa koetaan esiintyvän. Koska vastaajajoukon koko ei merkittävästi vaikuta tulosten käsittelyyn tarvittavaan työmäärään, kysely on kohdistettu yksittäistä projektia laajemmalle populaatiolle riittävän edustavan otannan varmistamiseksi. Strukturoidulla haastattelulla olisi mahdollisesti voitu saada enemmän ja tarkempaa tietoa, mutta suuren ihmismäärän haastattelu käytössä olevilla

resursseilla olisi käytännössä mahdotonta. Näin ollen riittävä tietoa saadaan kyselyn avulla kerätty tehokkaasti.

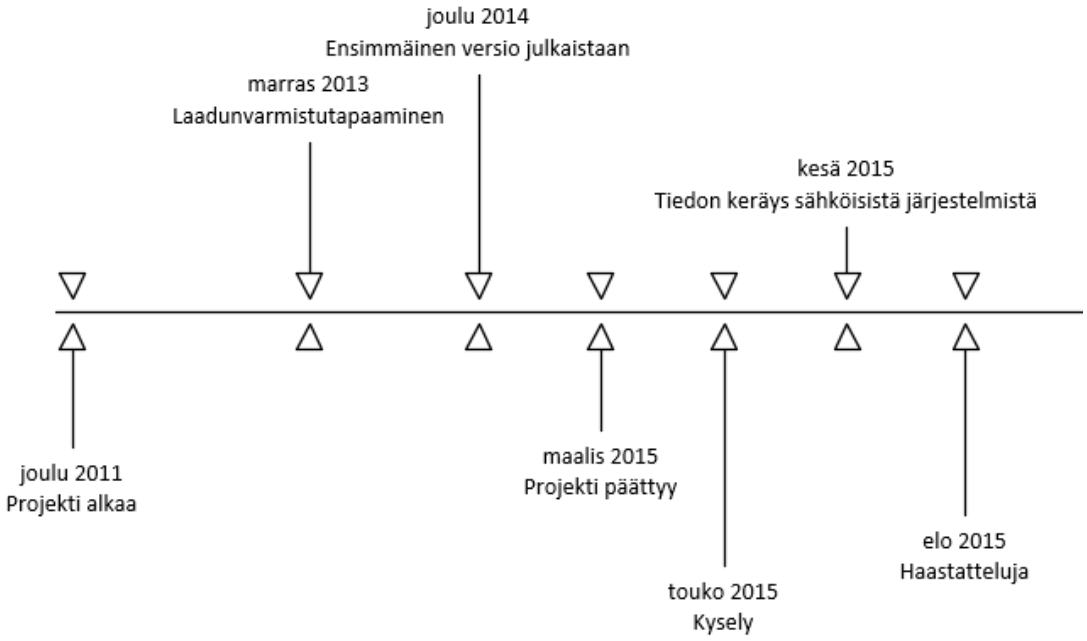
Näistä tiedoista pyritään havaitsemaan hukan määrää. Arvovirroista nähdään läpimenoaikojen sekä tahtiaikojen avulla hukan määrä työpäivissä, kvalitatiivisesta tiedosta saadaan viitteellistä tietoa mahdollisista hukista ja niiden määristä. Kyselyn tiedoista voidaan myös analysoida onko projekteihin osallistuvien eri ryhmien (kuten IT ja liiketoiminta) välillä eroja koetuissa hukissa. Mitattujen ja koettujen hukkien perusteella tunnistetaan suurimmat hukan lähteet sekä annetaan suosituksia niiden poistamiseksi.

4.3 *Tutkimusaineisto*

Tutkielman laatija keräsi tutkimusaineistoa osittain IT-yksikön laajuisesti ja asiakasosastoilta. Suurin osa aineistosta kerättiin kuitenkin yhdestä projektista, joka esiteltiin aliluvussa 3.3. Aineisto koostuu ohjelmistojärjestelmäprojektin tuotoksista: asiakasosastoilta sekä IT-yksiköltä laajemmin kerättiin tietoa kyselyllä sekä haastatteluilla.

Tutkimuksen tavoitteena on ohjelmistokehityksen tehokkuutta asiakkaan näkökulmasta. Kuten aiemmin on esitelty, tehokkuutta voidaan parantaa poistamalla kehitystyön arvovirrasta hukkaa. Arvovirrat on muodostettu laadunvarmistustapaamisen muistiinpanojen, projektinhallintaan käytetystä sähköisestä työtilan tietojen, projektitiimin jäsenille suoritettujen haastatteluiden sekä tutkielman laatijan henkilökohtaisista huomioiden avulla.

Tutkielmassa suoritettiin myös kysely, joka jaettiin IT-yksikön ohjelmistokehitysprojekteihin osallistuneille, niin IT-yksikössä kuin liiketoimintaosastoilla. Kyselyn tulosten, arvovirtojen sekä haastatteluiden avulla vastataan taulukossa 1 esiteltyihin tutkimuskysymykseen 1. **”*Mitkä olivat suurimmat hukkaa aiheuttaneet tekijät päättyneessä ohjelmistoprojekteissa*”** sekä apukysymykseen 2. **”*Miten hukan lähteiden poistoa voidaan priorisoida?*”**. Arvovirrat paljastavat suurimmat mitattavissa olevat hukan lähteet, ja haastatteluiden pyritään paikkaamaan mittauksissa esiintyviä aukkoja. Kysely paljastaa mitkä hukan lähteet koetaan subjektiivisesti suurimmiksi. Mikäli hukka on mitatusti suuri, ja koetaan suureksi, kannattaa sen poistoa priorisoida. Kuvassa 4 on annettu suuntaa antava kuvaus tiedon keräyksen aikataulusta suhteessa projektin aikatauluun.



Kuva 4: Tiedon keräyksen aikajana

4.3.1 Käyttötapausten arvovirrat

Projektin tuottaman järjestelmän eri ominaisuudet oli kirjattu ketterien käytäntöjen mukaisiksi käyttötapauksiksi [JSB11]. Joissain tapauksissa yksittäinen ominaisuus koostui useasta yksittäisestä käyttötapauksesta. Koska järjestelmä on hyvin laaja, yksittäisiä käyttötapauksia oli yhteensä 340, joista muodostui analysoinnin mahdollistavia kokonaisuuksia 102 kappaletta.

Läpimenoaika ja tahtiaika määriteltiin luvussa 2. Näiden käyttötapausten yhteydessä läpimenoajalla tarkoitetaan aikaa, joka kuluu käyttötapauksen luomisen aloittamisesta sen käyttöönottoon. Tahtiajalla tarkoitetaan työpäivien summaa, joka käytetään käyttötapauksen toteutuksen työstämiseen sekä toteutuksen käyttöönottoon.

Käytännön syistä jokaiselle yksittäiselle käyttötapaukselle ei laskettu tahtiaikoja ja läpimenoaikoja. Sen sijaan käyttötapaukset lajiteltiin kolmeen ryhmään niiden arvioidun työmäärän mukaan. Ryhmitys oli seuraava:

- Pienet: arvioitu työmäärä tasan 15 työpäivää tai alle (58 kpl)
- Keskikokoiset: arvioitu työmäärä yli 15 mutta alle 40 työpäivää (34 kpl)
- Suuret: arvioitu työmäärä tasan tai yli 40 työpäivää (10 kpl)

Työpäiviä olivat päivät maanantaista perjantaihin. Jokaisesta ryhmästä valittiin yksi

mahdollisimman monipuolisesti projektin kehitystyötä kuvaavaa tapaus. Pienistä valittiin käyttötapaus nimellä UC119, keskikokoisista UC172 ja suurista UC001 tarkempaa analyysiä varten.

Käyttötapauksien kaikkia työvaiheita ei merkitty sähköiseen työtilaan. Tästä syystä käyttötapauksien arvovirrasta ei voida saada täysin tarkkaa kuvaa, mutta kaikille valituille käyttötapauksille löytyi seuraavat mittauspisteet:

Käyttötapauksen luomisen aloituspäivämäärä

Aloituspäivämääräksi asetettiin käyttötapaukseen liittyvän sähköisen dokumentin luomispäivämäärä, jollei muuta päivämäärä noussut esiin.

Käyttötapauksen ensimmäisen version valmistuminen

Viimeinen päivämäärä, jolloin määrityksen sähköistä dokumenttia on muokattu ennen toteutuksen aloitusta.

Käyttötapaus otettu toteutukseen

Päivämäärä, jolloin ensimmäinen käyttötapaukseen liittyvä sähköisellä Scrum-taululla oleva tehtävä on otettu työn alle.

Käyttötapauksen toteutus valmis

Päivämäärä, jolloin käyttötapauksen viimeinen työtehtävää edustava kortti on merkitty valmiiksi.

Käyttötapauksen viimeinen versio valmis

Päivämäärä, jolloin dokumenttia on muokattu viimeisen kerran ennen järjestelmän tuotantoon ottoa.

Käyttötapauksen hyväksymistestaus aloitettu

Päivämäärä, jolloin käyttötapauksen toteutuksen ensimmäinen hyväksymistesti on aloitettu. Käyttötapauksia testattiin myös ennen tätä, kuten aiemmin projektin kulkua esiteltäessä mainittiin, mutta näistä ei ollut saatavilla merkintöjä ja niiden katsotaan olevan osa toteutusvaihetta.

Käyttötapaukseen liittyvä viimeinen virheraportti suljettu

Päivämäärä, jolloin käyttötapauksen viimeinen virheraportti on suljettu. Virheraportit suljettiin vain hyväksytyn testiajon jälkeen, joten tämän jälkeen käyttötapaus on hyväksytty ja valmis käyttöönottoa varten.

Näin ollen käyttötapauksien arvovirrasta tunnistettiin neljä suurinta työvaihetta, joille voitiin sähköisestä työtilasta löytää selkeitä aloitus- ja lopetuspäivämääriä: käyttötapausten luominen, toteutus, testaus ja hyväksyntä sekä käyttöönotto.

Käyttötapauksiin liittyvät tiedot kerättiin sähköisistä työtiloista projektin päätyttyä.

4.3.2 Haastattelut

Haastattelut olivat puolistrukturoituja ja niissä valittujen käyttötapausten laatijoilta sekä toteuttajilta kysyttiin muutamia kysymyksiä. Haastateltavan annettiin vastata vapaasti ja keskustelua jatkettiin, mikäli jotain tutkielman kannalta oleellista nousi esille. Tarkoituksena oli selvittää kuhunkin käyttötapaukseen liittyviä yksityiskohtia, jotka ei välttämättä ollut merkitty sähköiseen työtilaan. Tutkielman laatija haastatteli projektiin osallistuneita henkilöitä, ja heiltä kysyttiin seuraavia asioita:

- Esiintyikö käyttötapaukseen liittyen viivytyksiä? Kuinka pitkiä ne olivat?
- Esiintyikö käyttötapaukseen liittyen uudelleenopettelua? Kuinka paljon tämä vaikutti työskentelyn tehokkuuteen?
- Kuinka monta tekijää käyttötapaukseen osallistui? Siirrettiinkö työtehtäviä?
- Jäikö jotain työtehtäviä pitkäksi aikaa kesken?

Haastatteluilla pyrittiin siis tukemaan muusta aineistosta tehtyjä havaintoja, sekä mahdollisesti tunnistamaan hukkatekijöitä joita muusta aineistosta ei ole havaittu. Haastattelut suoritettiin projektin päätyttyä.

4.3.3 Kohdeprojektin puitteissa tehty laadunvarmistustapaaminen

Tarkastellun projektin aikana tehtiin yksi tapaaminen, jossa pohdittiin projektiin liittyviä ongelmia. Tapaamiseen kutsuttiin kaikki henkilöt, jotka projektissa olivat jollain tavalla mukana. Henkilöt jaettiin pienryhmiin, ja heille esiteltiin Leaniin liittyviä käsitteitä. Pienryhmiä pyydettiin keskustelemaan projektiin liittyvistä ongelmista, ja nostamaan esiin heidän mielestään tärkeimmät ongelmakohdat. Nämä kohdat pyrittiin sitten lajittelemaan TPS:n seitsemän hukan mukaiseen luokkaan, ja ongelmista sekä niiden ratkaisuista keskusteltiin koko tapaamiseen osallistuneen henkilöstön kesken.

Tapaamisen tuloksista tehtiin järjestäjien toimesta taulu, johon tulokset on koostettu. Taulu on tutkielman liitteenä numero 1. Havaitut hukkatekijät merkittiin tauluun tähdellä, ja siihen kirjoitettiin hukan luokka, hukan numero sekä koettu syy hukalle. Mitä suu-

rempi tähti on kooltaan, sitä vakavammaksi hukka koettiin.

Tätä aineistoa tarkasteltiin ohjelmistokehityksen hukkien näkökulmasta, ja tauluun kootut hukat luokiteltiin uudelleen aliluvussa 2.1.1. esiteltyjen hukkien mukaisesti. Hukkien frekvenssit laskettiin ja niiden koettu merkitys merkittiin taulukkoon. Näin muodostettua taulukkoa käytettiin tukemaan muuta aineistoa.

Tapaaminen järjestettiin projektin puolen välin jälkeen projektipäällikön toimesta.

4.3.4 Kysely koetusta hukasta

Tutkielman laatija lähetti kyselyn 58:lle IT- ja liiketoiminta-asiantuntijalle, jotka olivat osallistuneet organisaation eri IT-projekteihin. Nämä projektit kattoivat suuren osan toimijan liiketoiminta-alueista, ja tarjosivat näin kattavan otannan erilaisia näkökulmia kyselyä varten. Kysely suoritettiin kohteena olevan projektin päättymisen jälkeen.

Kyselyn alussa vastaajaa pyydettiin merkitsemään projektit, johon vastaaja on osallistunut. Näin tulosten tarkastelua voitiin tarvittaessa rajata vain tiettyihin projekteihin. Lisäksi vastaajaa pyydettiin merkitsemään roolit, joissa hän on toiminut projekteissa sekä merkitsemään yksi rooli, jossa hän on toiminut useimmin. Tämä tehtiin, koska tarkastelun kohteena olevassa organisaatiossa näkökulmat ja intressit ohjelmistokehitysprojekteihin ovat yleensä jaettavissa kahteen luokkaan; liiketoiminta ja IT. Kartoittamalla roolit pyrittiin tutkielmassa varmistamaan, että kyselyn vastaukset eivät heijasta vain toisen ryhmän intressejä. Vastaajat luokiteltiin käyttämällä roolia, jossa hän oli useimmiten toiminut.

Projekteista tunnistetut ja valittavissa olevat roolit olivat seuraavat:

- Testaaja
- Määrittelijä
- Testauspäällikkö
- Kehittäjä
- Admin
- Projektipäällikkö
- IT-koordinaattori

Näistä ensimmäiset kolme (testaaja, määrittelijä ja testauspäällikkö) luokiteltiin liiketoiminta-luokkaan ja loput neljä (kehittäjä, admin, projektipäällikkö ja IT-kordinaattori) IT-luokkaan.

Tämän jälkeen henkilöitä pyydettiin vastaamaan 21 kysymykseen, jotka oli laadittu sijoittumaan 7 ohjelmistokehitykseen mukautettuun Lean-hukan mukaiseen luokkaan. Kysymyksiin vastattiin Likertin asteikolla 1 - 5. Kysymykset pyrittiin asettamaan mahdollisimman negatiiviseksi, koska kyselyn tarkoituksena oli tuoda esiin ongelmakohtia. Lomakkeella kysymykset olivat satunnaisessa järjestyksessä. Kysymykset ja niiden arvoasteikot ovat tutkielman liitteessä numero 2.

4.3.5 Muut muistiinpanot ja havainnot

Koska tutkielman laatija työskenteli kyseisessä projektissa, muuna aineistona käytettiin laatijan omia havaintoja ja muistiinpanoja projektin ajalta.

5 Tutkimustulokset

Tässä luvussa käsitellään ensin koettua hukkaa koskevasta kyselystä ja laadunvarmistustapaamisen muistiinpanoista saadut tulokset. Näiden tarkoituksena on selvittää, kuinka paljon hukkaa projekteihin osallistuvat henkilöt kokevat projekteissa esiintyvän. Esimerkiksi jonkin tiedon puuttuminen saattaa pakottaa kehittäjän keskeyttämään meneillään olevan työtehtävän useiksi päiviksi, mutta kehittäjä voi sillä välin edistää jotain toista työtehtävää. Sen sijaan kehitysympäristön tuhoutuminen todennäköisesti estää minkään työtehtävän edistämisen tehokkaasti. Vaikka kehitysympäristön tuhoutuminen ei aiheuta kuin päivän viiveen, kun taas tiedon puuttuminen viivyttäisi projektia pitkällä aikavälillä useita päiviä, niin hukka voidaan kokea huomattavasti suuremmaksi kun mitään ei voida edistää. Näiden jälkeen käsitellään yksittäisitä käyttötapauksista saadut kvantitatiiviset ja kvalitatiiviset havainnot.

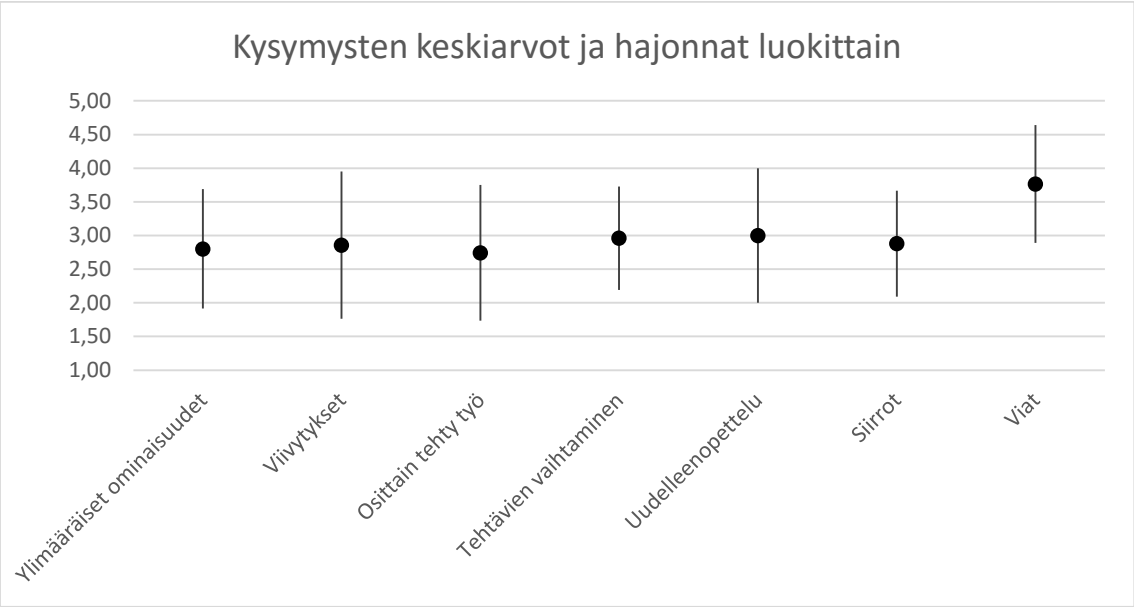
Kokonaisuutena tutkimustulosten tarkoituksena on auttaa vastamaan taulukossa 1 esiteltyyn tutkimuskysymykseen 1. ”**Mitkä olivat suurimmat hukkaa aiheuttaneet tekijät päättyneessä ohjelmistoprojekteissa?”**. Tutkimalla koettua hukkaa tässä tutkielmassa on pyritty löytämään ne hukan lähteet, jotka subjektiivisesti koetaan suurimmiksi. Jos suureksi hukaksi koettu asia voidaan myös kvantitatiivisella mittauksella osoittaa suureksi hukan lähteeksi, kyseinen asia kannattanee nostaa prosessin tehokkuuden parantamiseksi tehtävän muutoslistan kärkipäähän.

5.1 *Koettu hukka*

Tutkielmassa koettua hukkaa tutkittiin kyselylomakkeen avulla. Kyselylomaketta esiteltiin aliluvussa 4.2.4, ja sen kysymykset vastausarvoasteikkoineen ovat nähtävillä liitteessä 2. Kyselyn tulokset ovat tutkielman liitteessä numero 3.

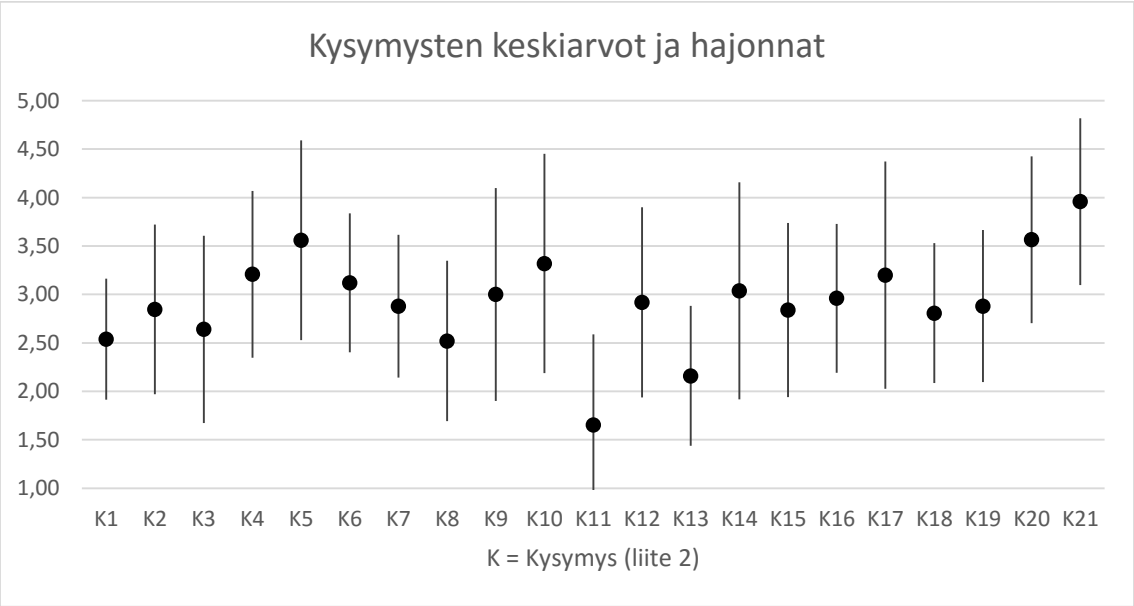
5.1.1 Kyselyn tulokset

Kysely lähetettiin 58 henkilölle, jotka osallistuivat toimijan IT-projekteihin eri rooleissa. Osallistujien roolit on esitelty aliluvussa 4.3.4. Kyselyyn vastasi 27 henkilöä, vastausprosentin ollessa näin noin 47 %.



Taulukko 4: Kyselyn tulosten keskiarvot hukkaluokkien mukaan

Kun kyselyn tuloksia tarkastellaan hukkaluokkien keskiarvojen tasolla (taulukko 4), organisaation IT-projekteissa on vastaajien mukaan havaittavissa jokaisen luokan mukaista hukkaa kohtalaisia määriä. Ylimääräisiä ominaisuuksista (keskiarvo 2,80) ja osittain tehdystä työstä (keskiarvo 2,74) johtuvaa hukkaa koettiin olevan hivenen vähemmän kuin muita, kun taas vioista johtuvaa hukkaa (keskiarvo 3,77) koettiin olevan selvästi muita hukkia enemmän.



Taulukko 5: Kyselyn tulosten keskiarvot kysymyksittäin

Kysymyksen numero (liite 2)	**Keskiarvo (liite 3)**
1	2,54
2	2,85
3	2,64
4	3,21
5	3,56
6	3,12
7	2,88
8	2,52
9	3
10	3,32
11	1,65
12	2,92
13	2,16
14	3,03
15	2,84
16	2,96
17	3,20
18	2,81
19	2,88
20	3,56
21	3,96

Taulukko 6: Kysymysten keskiarvot numeroina

Kun vastauksia tarkastellaan kysymyksittäin (taulukot 5 ja 6) voidaan huomata, että tuotteisiin koettiin toteutettavan ominaisuuksia ilman määrityksiä (K4) jonkin verran. Toteutumattomia tai käyttämättömiä ominaisuuksia koettiin olevan vähäisiä määriä (K1, K2) eikä pyytämättömiä ominaisuuksia juuri toteutettu palveluihin (K3).

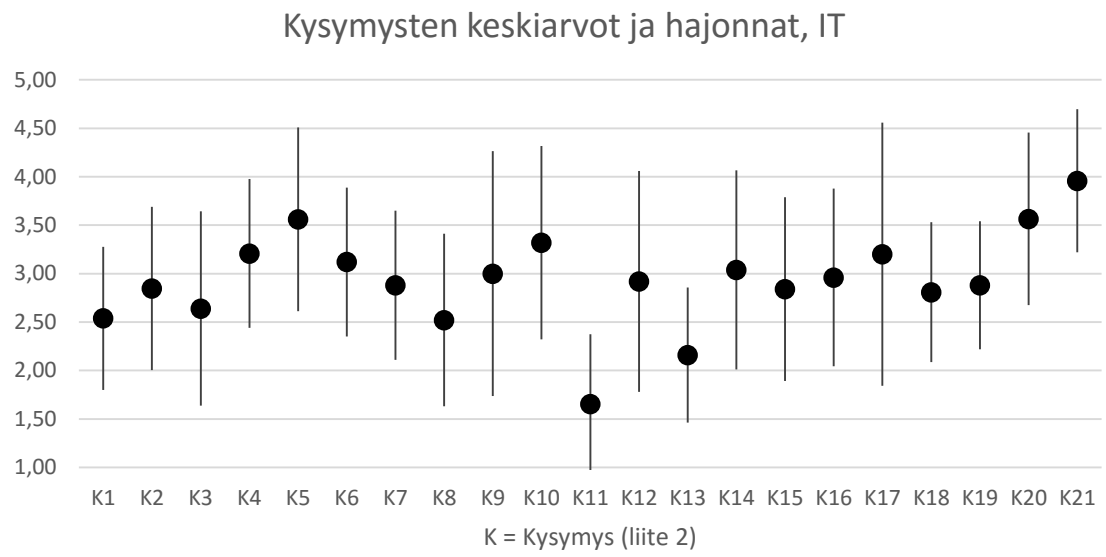
Viivytyksistä tiedon odotteluun koetaan kuluvan kohtalaisia määriä aikaa (K6) ja resurssien odotteluun vielä hieman enemmän (K5). Tarvittavan tiedon koettiin myös olevan vaikeasti saatavilla (K10). Muutosten tekemisen ei koettu olevan erityisen vaivalloista (K8) eikä tuotteiden testaukseen koettu käytettävän liikaa aikaa (K11).

Osittain tehdystä työstä määritysten ennenaikaista lukitsemista (K12), työtehtävien keskeytymistä (K14) ja tuotteiden kehittämistä epäluotettavan tiedon varassa (K15) koettiin esiintyvän jonkin verran. Myös tehtävien vaihtamista koettiin tapahtuvan jossain määrin (K16).

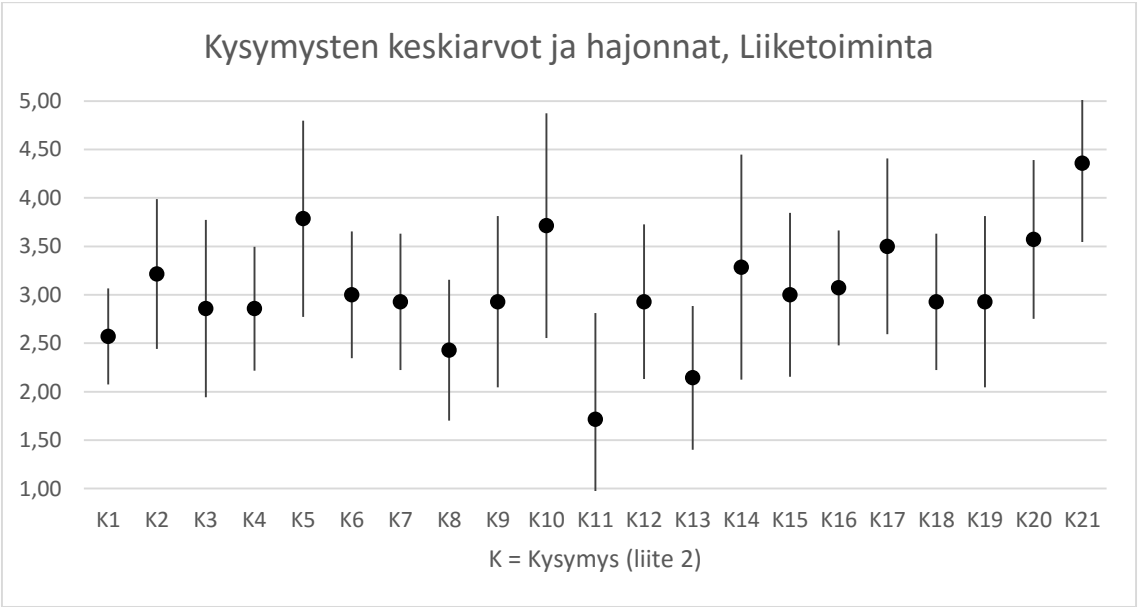
Tämä näkyy ehkä jossain määrin myös uudelleenopettelu-kategoriassa, sillä työtehtävien keskeytyksien koettiin aiheuttavan uudelleenopettelua kohtalaissa määrin (K17). Työtehtävien siirtoja henkilöltä toiselle koettiin tapahtuvan vähän tai kohtalaisesti (K19).

Tuotteissa koettiin esiintyvän paljon vikoja (K21), ja epäkypsien tai epäsopuisien tekniikoiden aiheuttavan kohtalaisesti tai paljon ongelmia (K20).

Kuten aiemmin mainittu, asiakasosastot ovat useissa projekteissa aktiivisesti mukana, yleensä vaatimusmääritysten ja testaamisen osalta. Koska suurimalla osalla liiketoiminnan edustajista ei ole IT-taustaa, ja myös yleensä käyttävät projektien tuottamia tuotteita itse, on mielenkiintoista tarkastella, miltä kehitysprosessien hukka näyttää heidän näkökulmastaan verrattuna kehittäjiin.



Taulukko 7: Kyselyn tulosten keskiarvot kysymyksittäin, vain IT-asiantuntijoiden vastaukset



Taulukko 8: Kyselyn tulosten keskiarvot kysymyksittäin, vain liiketoiminta-asiantuntijoiden vastaukset

Kysymyksen numero (liite 2)	**Keskiarvo IT (liite 3)**	**Keskiarvo Liiketoiminta (liite 3)**
1	2,62	2,57
2	2,54	3,21
3	2,62	2,86
4	3,85	2,86
5	3,15	3,79
6	3,15	3,00
7	2,85	2,93
8	2,77	2,43
9	3,31	2,93
10	3,08	3,71
11	1,69	1,71
12	3,08	2,93
13	2,23	2,14
14	2,85	3,29
15	2,85	3,00
16	2,92	3,07
17	3,00	3,50
18	2,69	2,93
19	2,85	2,93
20	3,77	3,57
21	3,62	4,36

Taulukko 9: Kysymysten keskiarvot IT ja liiketoiminta

Ensimmäinen ero merkittävä ero kysymyksen 2 kohdalla (taulukot 7, 8 ja 9). Kehittäjien mielestä tätä tapahtuu vain vähän tai kohtalaisesti, mutta liiketoiminnan edustajien mielestä tätä esiintyy kohtalaisesti tai paljon. Vastavuoroisesti kehittäjien mielestä tuottei-

siin toteutetaan paljon ominaisuuksia ilman määrityksiä (K4), kun taas liiketoiminnan edustajien mielestä vain vähän tai kohtalaisesti. Eli määrittelijät kokevat tekevänsä turhaa työtä määritellessään ominaisuuksia, joita ei toteuteta, mutta samaan aikaan kehittäjät kokevat että he joutuvat tekemään ominaisuuksia ilman määrityksiä.

Myös viivytyksien syyt nähdään poikkeavasti. Kummatkin kokevat, että resursseja odottelusta johtuvaa viivästystä esiintyy, mutta kehittäjät kokevat sen kohtalaisena (K5) kun taas liiketoiminta-puolen edustajat kokevat sitä esiintyvät paljon. liiketoiminta-puoli taas kokee, että projekteissa järjestetään turhia kokouksia vähän tai kohtalaisesti (K9), mutta kehittäjät kokivat, että niitä on kohtalaisesti tai paljon. Liiketoiminnan edustajat kokevat myös, että tuotekehitykseen tarvittava tieto on hyvin usein vaikeasti saatavilla (K10). Kehittäjät kokivat tätä huomattavasti vähemmän, mutta kuitenkin kohtalaisesti.

Osittain tehtyyn työhön liittyvää hukkaa koettiin melko samalla tavalla, mutta työtehtävien keskeytyksiä (K14) ja tuotteiden kehittämistä epäluotettavan tiedon varassa (K15) koettiin hieman vahvemmin kuin IT:n puolella. Uudelleenopettelu-luokassa työtehtävien keskeytyksistä johtuvaa uudelleenopettelua koettiin liiketoiminnan puolella esiintyvän kohtalaisesti tai paljon (K17) kun kehittäjät kokivat sitä kohtalaisesti. Myös olemassa olevien määritysten tai komponenttien hyväksikäyttö koettiin hieman eri tavalla (K18).

Suurin ero koetuissa hukissa näkyy vikojen kohdalla. Vaikkakin kehittäjät kokevat valmiissa tuotteissa esiintyvän kohtalaisesti tai paljon (K21), liiketoimintapuoli kokee niitä esiintyvän paljon tai jopa erittäin paljon.

5.2 *Laadunvarmistustapaamisen muistiinpanot*

Liitteenä 1 on kuva tapaamisen pienryhmissä tehdyistä havainnoista, ja tapaamista on kuvailtu tarkemmin aliluvussa 4.2.3.. Havaintoja saatiin seitsemästä hukka-kategoriasta: varastot, ylituotanto, ylimääräinen käsittely, tarpeeton liike, odottelu, viat ja käyttämättä jätetty työntekijöiden luovuus. Jotta näitä havaintoja voidaan hyödyntää, täytyy ne kartoittaa vastaamaan muiden havaintojen luokituksia. Tämä voidaan tehdä Poppendieckien [Pop03] mukaan tehdä Lean-tuotannossa tunnistetuille seitsemälle hukalle taulukossa 10 esitetyllä tavalla.

Lean-tuotannon hukat	Lean-ohjelmistokehityksen hukat
Varastot	Osittain tehty työ
Ylituotanto	Ylimääräiset ominaisuudet
Ylimääräinen käsittely	Uudelleenopettelu
Kuljetus	Siirrot
Tarpeeton liike	Tehtävien vaihtaminen
Odottelu	Viivytykset
Viat	Viat

Taulukko 5: Lean-tuotannon ja Lean-ohjelmistokehityksen hukkien vastaavuus

Muistiinpanoissa on myös merkintä Käyttämättä jätetty työntekijöiden luovuus-hukasta. Tälle ei kuitenkaan löydy vastaavuutta Lean-ohjelmistokehityksen hukista, joten sitä ei käsitellä tarkemmin. Tästä hukasta on myös vain yksi maininta, joka liittyy vahvasti ylimääräiseen käsittelyyn, joten tämän puutteen ei katsota olevan merkittävä.

Varastot – osittain tehty työ

Varastoista oli yksi maininta, määritykset (käyttötapaukset) odottamassa toteutuksen alkua. Näitä syntyi, koska kaikki käyttötapaukset kirjoitettiin valmiiksi ennen kuin yhtäkään otettiin toteutukseen.

Ylituotanto – ylimääräiset ominaisuudet

Kuten liitteessä 2 olevasta kuvasta nähdään, tässä kategoriassa oli kolme merkintää. Ylituotantoa koettiin olevan määritykset (käyttötapaukset), jotka odottavat toteutuksen aloitusta, toteutuksen tekeminen ilman määrityksiä sekä liian aikainen koulutus. Toteutusta odottavien käyttötapausten koettiin olevan hukkaa, koska tehdyt käyttötapaukset saattaisivat jäädä toteuttamatta, joten niihin käytetty työ olisi siis hukkaa. Toteutuksen tekeminen ilman määritystä taas ilmeni niin, että joissain tapauksissa kehittäjät ennakoivat käyttäjien tarpeita ja tekivät järjestelmään ominaisuuksia joita ei ollut määritelty.

Liian aikaisen koulutuksen pelättiin valuvan hukkaan, mikäli koulutukseen aikasin osallistuneet unohtavat koulutuksen ennen järjestelmän käyttöönottoa. Tämä ei kuitenkaan aivan sovi ylimääräisten ominaisuuksien kategoriaan, ja siitä on maininta odottelun alla, joten sen katsotaan tässä tutkielmassa kuuluvan sinne.

Tarpeeton liike – tehtävien vaihtaminen

Tässä kategoriassa oli kuusi mainintaa: tuplavirheraportit (muistiinpanoissa ”tupla-bugit”), korjausten puuttuminen testattavasta versiosta, saman työn tekeminen useaan kertaan sekä täsmennystarpeet toteutuksessa, liiketoiminnan alustavissa testeissä, kuin myös liiketoiminnan integraatiotesteissä. tuplavirheraportit tarkoitettiin sitä, että eri testaajat kirjasivat sähköiseen työtilaan saman vian useampaan kertaan, hieman eri kuvauksella. Tämä aiheutti turhaa analysointia jo korjatusta ongelmasta.

Korjausten puuttuminen testattavasta versiosta koettiin merkittäväksi ongelmaksi. Kun toteutuksen virheitä korjattiin, korjauksen tekijä vei korjatun koodin projektin yhteiseen versiohallintajärjestelmään. Tätä järjestelmää hyväksikäyttäen korjauksia vietiin erikseen sovittuina ajankohtina testattaviksi erilliseen testiympäristöön. Joissain tapauksissa korjattua koodia ei muistettu laittaa järjestelmään, tai se jäi jostain muusta syystä pois testattavaksi vietävästä versiosta. Virheeseen liitetty työtehtävä oli kuitenkin suljettu, joten sen oletettiin olevan mukana uudessa versiossa. Näin ollen testattavasta versiosta saattoi puuttua korjauksia. Kun testaaja ajoi testejä, ja havaitsi saman ongelman joka oli jo merkitty korjatuksi, hän ei aina voinut olla varma johtuko tämä siitä, että virhe ei ollut aidosti korjaantunut, vai puuttuiko koko korjaus testattavasta versiosta. Tämä aiheutti ylimääräistä analysointia niin kehittäjien kuin testaajienkin puolella.

Saman työn tekemistä useaan kertaan havaittiin kehittäjien keskuudessa. Esimerkiksi samoja aputoimintoja tai käyttöliittymän näkymiä tehtiin useita kertoja eri kehittäjien toimesta, koska kehittäjät eivät olleet tietosia aiemmin version olemassaolosta.

Testien ja toteutuksen täsmennystarpeilla tarkoitetaan sitä, että käyttötapauksia jouduttiin monessa tapauksessa muuttamaan ensimmäisen toteutuksen testien yhteydessä. Koska käyttötapausta muutettiin, myös testejä tuli muuttaa.

Tämän kategorian havainnoista mikään ei ole tehtävien vaihtamista. Tuplavirheraportit ja korjausten puuttuminen ovat vikoihin liittyviä asioita, saman työn tekeminen uudelleenopettelua ja täsmennystarpeet osittain tehtyä työtä.

Odottelu - viivytykset

Odottelusta oli kolme mainintaa. Yksi näistä oli testauksen aloittamiseen liittyvää odottelua. Joissain tapauksissa valmis toteutus odotti pitkiä aikoja ennen kuin sen testaus alkoi. Tässä välissä kehittäjä oli jo siirtynyt toisen tehtävän pariin, ja jos testauksessa paljastui ongelmia, kehittäjän aikaa kului toteutukseen liittyvien yksityiskohtien mieleen

palauttamiseen. Toinen maininta oli jo ylituotannon kohdalla mainittu liian aikainen koulutus. Näistä merkittävimmäksi nostettiin kuitenkin käyttötapausten täsmennystarpeiden havaitseminen vasta ensimmäisen version toteutuksen jälkeen. Näin ollen kehittäjät joutuivat odottamaan muutosten valmistumista tai siirtymään toisen tehtävän pariin.

Viat

Viat mainittiin yhteensä kuudessa kohdassa. Vikoja nähtiin toteutuksen virheissä, versioiden hallinnassa, korjauksien verifioinnissa, toteutuksen aikaisen testauksen puutteessa ja tarvittavien dokumentti hajanaisessa sijoittelussa sekä niiden ajantasaisuudessa.

Testien aikana toteutuksessa havaittiin yhteensä yli 3300 virhettä. Virheiden korjaus ja testaamisen koettiin aiheuttavan huomattavia määriä ylimääräistä työtä. Version hallinnalla viitataan samaan ilmiöön, jota esiintyi korjausten puuttumisen kohdalla; joissain tapauksissa ympäristöstä toiseen siirretystä uudesta versiosta puuttui osia. Tämä sama ongelma on myös korjauksien verifointien takana; testaaja ei voinut olla varma eikö vika ollut korjaantunut vai oliko korjaus jäänyt pois versiosta.

Käyttötapauksien toteutuksia oli tarkoitus testata itse myös itse toteutustyön aikana, mutta monessa tapauksessa näin ei kuitenkaan tehty. Tämä johti siihen, että nopeasti havaittuna helposti korjattavat ongelmat nousivat esiin vasta myöhemmin, kun toteutustyö oli jo edennyt pidemmälle, näin huomattavasti vaikeuttaen korjausta.

Taulukossa 12 on yhteenveto laadunvarmistustapaamisen tuloksista.

Hukan tyyppi	**Mainintojen frekvenssi (liite 2)**	**Koettu vakavuus**
Osittain tehty työ	3	Kohtalainen
Ylimääräiset ominaisuudet	3	Kohtalainen
Uudelleenopettelu	1	Kohtalainen
Siirrot	0	Ei havaittu
Tehtävien vaihtaminen	0	Ei havaittu
Viivytykset	3	Merkittävä
Viat	10	Kohtalainen

Taulukko 11: Laadunvarmistustapaamisen muistiinpanojen havaintojen yhteenveto

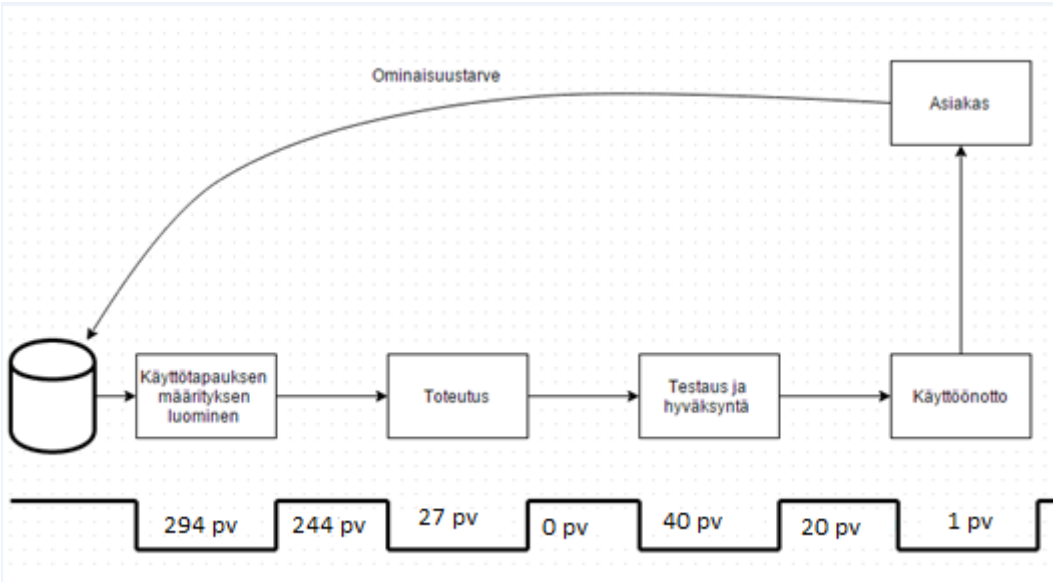
5.3 *Käyttötapausten mitattu hukka*

Valituille käyttötapauksille tutkielman laatija laski jokaiselle käyttötapaukselle tahtiajat ja läpimenoajat. Nämä on merkitty jokaisen käyttötapauksen kohdalle käyttötapauksen arvovirtaa kuvaaviin kuviin 5, 6 ja 7. Sähköisten työtilojen puutteellisia merkintöjä voitiin osin korjata haastattelemalla käyttötapausten parissa työskennelleitä henkilöitä mahdollisten hukkien paljastamiseksi

5.3.1 Käyttötapaus 1 (pieni)

Käyttötapaus liittyi järjestelmää käyttävien organisaatioiden laskutukseen. Käyttötapauksen kirjoitti yksi määrittelijä, ja sen toteutti yksi kehittäjä. Käyttötapaus oli haastateltujen mukaan niin teknisesti kuin liiketoiminnallisesti suoraviivainen. Ainoastaan laskutukseen liittyvät säännöt olivat monimutkaisia. Käyttötapauksen tekniseen toteutukseen oli kehittäjien toimesta arvioitu aikaisemman kokemuksen perusteella menevän 15 työpäivää.

Arvovirta



Kuva 5: Käyttötapauksen 1 arvovirtakartta

Eri vaiheisiin käytettyjen työpäivien määrä on merkitty kuvaan 5. Käyttötapauksen luominen aloitettiin sähköisen työtilan merkintöjen mukaan 13.7.2012 ja ensimmäinen, toteutukseen hyväksytty versio valmistui 28.8.2013. Näin ollen käyttötapauksen luomiseen meni 294 työpäivää. Käyttötapauksen toteutus aloitettiin 4.8.2014 eli aloituksen ja käyttötapauksen valmistumisen välissä oli 244 työpäivää. Toteutuksen viimeinen tehtävä valmistui 9.9.2014, joten itse toteutukseen käytettiin 27 työpäivää.

Sähköisen työtilan mukaan käyttötapauksen testaus aloitettiin 30.7.2014, eli kaksi työpäivää ennen kuin toteutusta edes aloitettiin. Tähän voi olla selityksenä, että kehittäjä on jo tehnyt toteutusta ennen kuin merkitsi työtehtävän olevan työn alla. Kysyttäessä kehittäjä ei tällaista kuitenkaan muista tapahtuneen. Käyttötapauksessa testausta ja toteutusta tehtiin jonkin aikaa samanaikaisesti.

Koko testausvaiheen aikana käyttötapauksesta kirjattiin 19 vikaraporttia, ja se oli testaus-ja-korjaus syklissä toteutusvaiheen päätyttyä vielä 41 työpäivää, ennen kuin viimeinen virheraportti suljettiin (eli käyttötapaus läpäisi hyväksymistestit) 4.11.2014. Yksi testi saattoi kestää minuutteja ajaa, mutta jos testi ei mennyt läpi, se odotti useamman päivän jotta se voitiin ajaa uudelleen. Näin ollen testaukseen ja hyväksyntään käytettiin arviolta yksi työpäivä kokonaisuudessaan, hajautettuna testausvaiheen ajanjaksolle. Tästä johtuen voidaan 40 työpäivää tästä ajasta olleen vioista johtuvaa hukkaa. Käyttötapaus otettiin käyttöön järjestelmän 1. version julkaisun myötä 1.12.2014, joten testausvaiheen ja käyttöönoton välillä kului 20 työpäivää.

Käyttötapauksen läpimenoaika oli 626 työpäivää ja tahtiaika 28 työpäivää. Kuten aliluvussa 2.3. esiteltiin, tahtiajalla tarkoitetaan sitä aikaa, joka käytettiin tuotteen aktiiviseen tuottamiseen. Tässä tapauksessa tahtiaikaan laskettiin toteutukseen käytetty aika (27 työpäivää) sekä testaukseen käytetty aika (1 työpäivä).

Haastattelut

Haastattelussa määrittelijä mainitsee, että hän ei pystynyt täysin keskittymään käyttötapauksen kirjoittamiseen, vaan työ keskeytyi jatkuvasti ja hän joutui siirtymään jonkin muun tehtävän pariin:

> ”Kylhä mä testasin ja kaikkee samaan aikaan. Mä en ikinä pystyny, et teen nyt pelkästään Use Caseia. Vaik sellasta oli suunnitteilla, mut käytännössä ku meil oli koko ajan testei päällä. Ja mä olin niinku testimanageri, mult tultiin koko ajan kysyy jotain.”
>
> - Käyttötapauksen 1 määrittelijä

Tässä tapauksessa tehtävien vaihto johtui siis määrittelijän kahdesta roolista. Määrittelijä mainitsee myös, että nämä keskeytykset aiheuttivat myös hieman uudelleenopettelua, mutta haitan olleen pieni:

”…(käyttötapaukseen liittyviä laskukaavoja) kyl mä jouduin niit niiku (uudelleenopettelee). Jos tuli joku tauko tai keskeytys, ni joutu lähtee aina alusta hakee et miten tää laskukaava nyt menikää.”

- Käyttötapauksen 1 määrittelijä

Määrittelijän mukaan hän pystyi kuitenkin kirjoittamaan käyttötapauksen ilman muuta odottelua tai viivästyksiä.

Toteutuksesta vastannut kehittäjän mukaan toteutuksessa ei ilmennyt suurempia ongelmia, ja hän pystyi tekemään sen alusta loppuun ilman merkittäviä keskeytyksiä. Kehittäjän ja määrittelijän mielestä käyttötapauksessa ei myöskään ollut ylimääräisiä ominaisuuksia.

Käyttötapauksessa havaittu hukka

Kuten määrittelijän haastattelusta ilmenee, käyttötapauksen kirjoittamiseen liittyi hukkaa uudelleenopettelun ja tehtävien vaihtamisen muodossa. Näitä hukkia ei pystytty kvantitatiivisesti mittamaan, mutta haastateltavan mukaan se oli vähäistä. Merkittävämpää hukkaa ilmenee kuitenkin käyttötapauksen valmistumisen ja toteutuksen aloituksen välissä.

Valmis käyttötapaus odotti toteutuksen aloitusta 244 päivää. Tämä on suoraan turhaa viivästystä, ja sen voidaan katsoa olevan myös osittain tehtyä työtä, sillä käyttötapaus odotti ”varastossa” käsittelyä. Tämä saattoi johtaa myös uudelleenopetteluun, kun määrittelijä palautteli toteutuksen alkaessa käyttötapausta mieleensä näin pitkän ajanjakson jälkeen. Osittain tehdystä työstä tai uudelleenopettelusta johtuvaa hukkaa ei kuitenkaan voitu mitata työpäivissä.

Hukkaa voitiin mitata niin vikojen muodossa, kuin käyttöönoton odottelussa. Vikojen löytämiseen ja korjaamiseen kului 40 työpäivää. Käyttöönoton odottelu taas aiheutti viivytyksen muodossa 20 työpäivää hukkaa.

Tässä käyttötapauksessa merkittävimmät hukat syntyivät siis viivytyksistä ja vioista.

Kun lasketaan yhteen käyttöönoton odottelusta (20 työpäivää), toteutuksen aloituksen odotuksesta (244 työpäivää) sekä vikojen korjaamiseen käytetystä ajasta (40 työpäivää) johtuvat hukat, saadaan hukan osuudeksi käytetystä kokonaisajasta (622 työpäivää)

$$\frac{20 + 244 + 40}{622} \approx 0{,}48$$

eli yli 48 % ajasta oli hukkaa. Eroteltuna viivytysten osuus oli

$$\frac{20 + 244}{622} \approx 0{,}42$$

eli noin 42 % kokonaisajasta ja vikojen

$$\frac{40}{622} \approx 0{,}06$$

eli noin 6 % kokonaisajasta.

Arvovirran tehokkuus saadaan laskemalla yhteen toteutukseen ja testaukseen käytetyt työpäivät ja jakamalla se käyttötapaukseen käytettyjen työpäivien määrällä. Arvovirran tehokkuus oli siis

$$\frac{27 + 1}{622} \approx 0{,}05$$

eli noin 5 %. Hukkien yhteenveto on esitelty taulukossa 12.

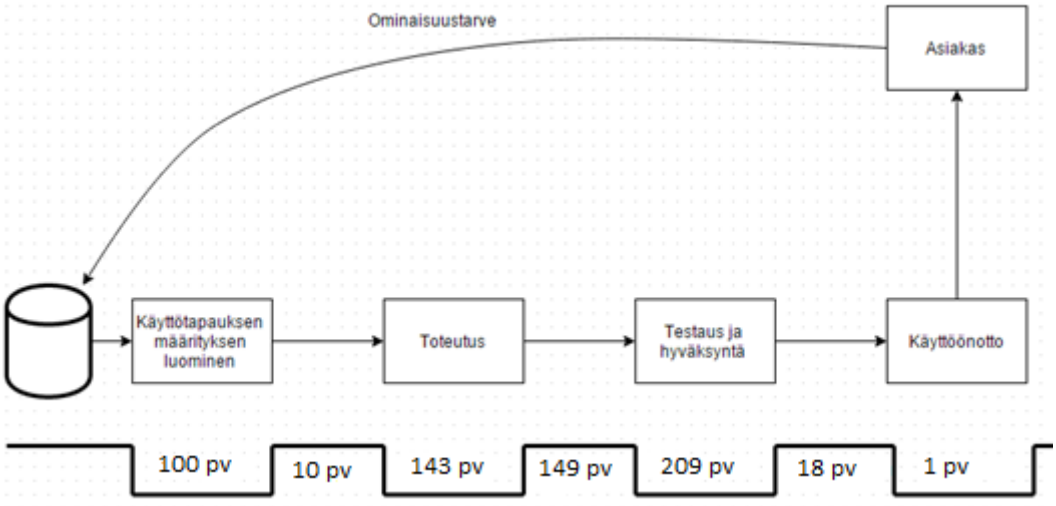
Hukan tyyppi (Poppendieck [Pop03])	**Millä tavalla havaittu**	**Mitattu hukka (jos laskettavissa) työpäivinä**
Osittain tehty työ	Haastatteluista	
Ylimääräiset ominaisuudet	Ei havaittu	
Uudelleenopettelu	Haastatteluista	
Siirrot	Ei havaittu	
Tehtävien vaihtaminen	Haastatteluista	
Viivytykset	Haastatteluista ja sähköisestä aineistosta	20 + 244
Viat	Haastatteluista ja sähköisestä aineistosta	40

Taulukko 12: Käyttötapauksesta 1 mitattujen hukkien yhteenveto

5.3.2 Käyttötapaus 2 (keskisuuri)

Käyttötapauksen tarkoituksena oli tallentaa ulkopuolelta viestien muodossa tulevaa varallisuuden hallintaan liittyvää tietoa järjestelmään. Tämä toiminnollisuus oli keskeinen osa järjestelmää, ja sen toteutuksen kestoksi oli kehittäjien toimesta arvioitu aikaisemman kokemuksen perusteella 29 työpäivää. Toteutukseen osallistui kaksi kehittäjää, joiden työtehtävät oli jaettu selkeästi erillisiksi kokonaisuuksiksi. Käyttötapauksen kirjoitti yksi määrittelijä.

Arvovirta



Kuva 6: Käyttötapauksen 2 arvovirtakartta

Eri vaiheisiin käytettyjen työpäivien määrä on merkitty kuvaan 6. Käyttötapauksen kirjoittaminen aloitettiin 11.7.2012 ja sen ensimmäinen, toteutukseen hyväksytty versio valmistui 100 työpäivää myöhemmin, 27.11.2012. Sen toteutus aloitettiin suhteellisen nopeasti version valmistuttua, 10.12.2012 eli 10 työpäivää myöhemmin. Käyttötapaus kävi läpi useita iteraatioita toteutuksen aikana, ja viimeinen työtehtävä merkittiin sähköiseen työtilaan valmiiksi 26.6.2013 kehittäjän toimesta. Näin ollen toteutukseen käytettiin 143 työpäivää.

Toteutuksen testaus alkoi 20.1.2014, joten sen ja toteutuksen valmistumisen välillä kului 149 työpäivää. Testauksen aikana käyttötapauksesta kirjattiin 34 virheraporttia ja näistä viimeinen suljettiin hyväksytysti korjattuna 6.11.2014 liiketoiminnan testaajien toimesta. Myös tässä käyttötapauksessa testit olisi voitu ajaa läpi yhdessä työpäivässä, joten vikojen korjaukseen käytettiin siis 209 työpäivää. Kuten käyttötapauksen 1 kohdalla, myös tässä testaukseen käytettiin noin 1 työpäivä yhteensä, joten 208 työpäivää voidaan

katsoa olevan vioista johtuvaa hukkaa. Käyttötapaus otettiin käyttöön järjestelmän 1. version julkaisun myötä 1.12.2014, joten testausvaiheen ja käyttöönoton välillä kului 18 työpäivää.

Käyttötapauksen läpimenoaika oli 630 työpäivää ja sen tahtiaika 144 työpäivää.

Haastattelut

Haastattelussa määrittelijä mainitsee, että käyttötapausta kirjoitettaessa hän on joutunut odottamaan lisätietoja organisaation ulkopuoliselta tietolähteeltä. Hän pystyi sähköpostiteistaan osoittamaan, että vastauksia odotettiin ensin 18.3.2013 – 12.4.2013 eli 20 työpäivää, ja toisella kerralla 4.3.2013 – 10.3.2013 eli 5 työpäivää lisää. Nämä keskeytykset eivät kuitenkaan ole haastateltavan mielestä johtaneet uudelleenopetteluun

> ”…ei sellasesta johtuen, et mä oon niiku joutunu odottamaan joltain jotain vastausta, niin en siitä johtuen oo joutunu tekee sellasta (palauttamaan mieleen tai uudelleenopettelemaan) ainakaan mitenkään merkittävästi”
>
> - Käyttötapauksen 2 määrittelijä

Määrittelijä pystyi myös odottelun aikana edistämään muita käyttötapaukseen liittyviä asioita, eikä odotus sinällään mainittavasti hidastanut käyttötapauksen valmistumista. Näin ollen odotusaikoja ei lasketa suoraan hukaksi. Määrittelijä sanoo, että pystyi tekemään tämän käyttötapauksen alusta loppuun ilman päällekkäisiä tehtäviä tai merkittäviä häiriöitä.

Kehittäjien mukaan toteutus sujui hyvin, eikä mitään huomattavia viivytyksiä tai muita ongelmia tullut haastateltavien mieleen.

Käyttötapauksessa havaittu hukka

Kun käyttötapaus oli kirjoitettu, syntyi toteutuksen aloituksen odottelusta 10 työpäivän verran hukkaa. Kuten aiemmassa käyttötapauksessa, myös tähän liittyi odottelun lisäksi osittain tehdyn työn hukkaa, sillä valmis käyttötapaus odotti jatkokäsittelyä. Edellisestä käyttötapauksesta poiketen, odotusaika oli kuitenkin hyvin lyhyt, eikä todennäköisesti aiheuttanut uudelleenopettelua.

Viivytykset aiheuttivat lisää hukkaa, kun toteutuksen testaamisen aloittamiseen kului 149 työpäivää ja käyttöönoton odotteluun 18 työpäivää. Testaus-ja-korjaussykleihin kului vuorostaan 208 työpäivää. Tässäkin käyttötapauksessa merkittävimmät hukat syn-

tyivät viivytyksistä sekä vioista. Kun lasketaan yhteen toteutuksen aloituksen odotuksesta (10 työpäivää), testaamisen aloittamisen odottelusta (149 työpäivää), käyttöönoton odottelusta (18 työpäivää) sekä vikojen korjaamiseen käytetystä ajasta (208 työpäivää) johtuvat hukat, saadaan hukan osuudeksi käytetystä kokonaisajasta (624 työpäivää)

$$\frac{10 + 149 + 18 + 208}{624} \approx 0{,}62$$

eli noin 62 % ajasta oli hukkaa. Eroteltuna viivytysten osuus oli

$$\frac{10 + 149 + 18}{624} \approx 0{,}28$$

eli noin 28 % kokonaisajasta ja vikojen

$$\frac{208}{624} \approx 0{,}33$$

eli noin 33 % kokonaisajasta.

Arvovirran tehokkuus oli (toteutukseen käytetyt työpäivät lisättynä testaukseen käytetyt työpäivät jaettuna kokonaisajalla)

$$\frac{143 + 1}{624} \approx 0{,}23$$

eli noin 23 %. Hukkien yhteenveto on esitelty taulukossa 13.

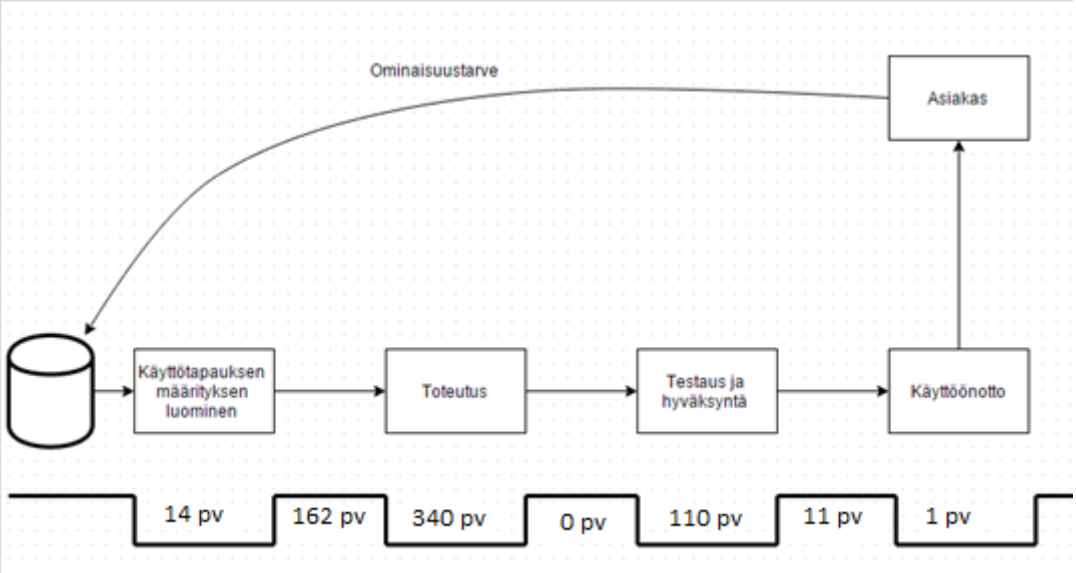
Hukan tyyppi	**Millä tavalla havaittu**	**Mitattu hukka (jos laskettavissa) työpäivinä**
Osittain tehty työ	Haastatteluista	
Ylimääräiset ominaisuudet	Ei havaittu	
Uudelleenopettelu	Ei havaittu	
Siirrot	Ei havaittu	
Tehtävien vaihtaminen	Ei havaittu	
Viivytykset	Haastatteluista ja sähköisestä aineistosta	10 + 149 + 18
Viat	Haastatteluista ja sähköisestä aineistosta	208

Taulukko 13: Käyttötapauksesta 2 mitattujen hukkien yhteenveto

5.3.3 Käyttötapaus 3 (suuri)

Tässä käyttötapauksessa kuvattiin uuden varallisuuden merkitsemistä järjestelmään viestien avulla. Tämä käyttötapaus oli eräs monimutkaisimmista ja tärkeimmistä käyttötapauksista koko järjestelmässä. Käyttötapauksen kirjoittamiseen osallistui kolme määrittelijää, ja sen toteutukseen kuusi kehittäjää. Sen toteutukseen arvioitiin menevän yhteensä 72 työpäivää.

Arvovirta



Kuva 7: Käyttötapauksen 3 arvovirtakartta

Tämän käyttötapauksen luominen aloitettiin 26.6.2012. Sen ensimmäinen, toteutukseen hyväksytty versio valmistui 13.7.2012 eli 14 työpäivää myöhemmin. Käyttötapaus valmistui kahteen aikaisempaan käyttötapaukseen verrattuna nopeasti. Se otettiin toteutukseen kuitenkin vasta 162 työpäivän päästä, 25.2.2013.

Käyttötapaus eli voimakkaasti toteutuksen aikana, ja siitä tehtiin 5 versiota, kunnes sen viimeinen versio valmistui 8.5.2014. Toteutuksen viimeinen tehtävä taas suljettiin 15.6.2014 kehittäjien toimesta. Näin ollen toteutukseen käytettiin sähköisen työtilan tietojen mukaan kokonaisuudessaan 340 työpäivää.

Toteutuksen testaus alkoi ensimmäisen kerran 22.8.2013, 212 työpäivää ennen viimeisen tehtävän sulkemista. Käyttötapaukseen liitettiin 114 virheraporttia, joten käyttötapauksen määritys, testaus ja toteutus elivät samanaikaisesti. Viimeinen virheraportti suljettiin hyväksytyn testikierroksen jälkeen 17.11.2014, 111 päivää toteutuksen viimeisen tehtävän sulkeuduttua liiketoiminnan testaajien toimesta. Kuten aiemmissa käyttötapauksissa, tässäkin testaukseen kului noin 1 työpäivä, eli 110 päivää kului virheiden

korjaamiseen. Valmis käyttötapaus odotti vielä 1.12.2014 tapahtunutta käyttöönottoa 11 työpäivää.

Käyttötapauksen läpimenoaika oli 638 ja sen tahtiaika 341 työpäivää.

Haastattelut

Käyttötapauksen 3 määrittelijästä vain yksi oli saatavilla haastattelua varten. Tämä määrittelijä oli kuitenkin kirjoittanut suurimman osan käyttötapauksesta ja haastatteluhetkellä vastasi käyttötapauksesta kokonaisuutena. Määrittelijä mainitsee useita päällekkäisiä työtehtäviä kirjoittamisen aikana:

> ”Mulla oli useampia use caseja (käyttötapauksia) samaan aikaan työstettävänä, mutta siis kun teki niitä use caseja niin keskitty vaan yhteen”
>
> - Käyttötapauksen 3 määrittelijä

Hän kertoi, että usein käyttötapauksen kirjoittaminen kuitenkin keskeytyi, koska hänen piti hakea tietoa muilta, eikä kirjottamista voinut jatkaa muutamaan päivään. Hän ei kuitenkaan pystynyt antamaan tarkkoja ajanjaksoja koska työ oli keskeytynyt. Keskeytyksen kuitenkin johtivat uudelleenopetteluun:

> ”…sit kun palas siihen use caseen, jota oli tehny pari päivää sitten, niin kyl sitä joutu aina palauttelee mieleen, et mitäs täs nyt sit oli tehty”
>
> - Käyttötapauksen 3 määrittelijä

Käyttötapauksen oli aloittanut yksi määrittelijä, jolta se siirtyi toiselle henkilölle. Toinen määrittelijä jäi kuitenkin kesken projektin eläkkeelle, ja vastuu käyttötapauksesta siirtyi kolmannelle määrittelijälle. Haastateltavan mukaan myös tämä johti uudelleenopetteluun, ja hän joutui myös korjailemaan osittain valmista käyttötapausta.

Käyttötapauksen toteutukseen osallistui 6 kehittäjää, joista 5 oli organisaation ulkopuolisia konsultteja ja yksi oma työntekijä. Vain organisaation oma työntekijä oli saatavilla haastatteluun. Kehittäjä sanoo, että toteutuksen yhteydessä ei ilmenny erityisiä viivästyksiä, mutta työtehtävien välillä jouduttiin hyppimään paljonkin ja tämä aiheutti uudelleenopettelua:

> ”…niin siinä tuli kyllä sitä, että sitä tavallaan sen workflown (järjestelmässä käytetty orkestraatiotekniikka), joka oli yks monimutkasimmista workflowsta,

niin mites tää menikään. Et tämmöstä oli paljon, koska siihen on jouduttu palaamaan monet kerrat.”

- Käyttötapauksen 3 kehittäjä

Kehittäjä myös mainitsee, että monet osat toteutuksesta tehtiin puolittain valmiiksi, ja viimeisteltiin myöhemmin:

”…ne oli toteutettu vähän sillai puolittain, että joku alternative flow oli kokonaan tekemättä, ja sit luultiin että se on kunnossa…useimmiten tää johtu siitä, että siinä speksissä (käyttötapaus) oli vähän semmosta epämääräsyyttä, joka johti sit siihen, että sitä (toteutuksen osaa) sit oltu tehtykkään.”

- Käyttötapauksen 3 kehittäjä

Koska tekijöitä oli useita, ja osa työstä oli keskeneräistä, tehtäviä myös siirreltiin haastateltavan mukaan tekijältä toiselle jatkuvasti.

Käyttötapauksessa havaittu hukka

Käyttötapauksen laatijan haastattelusta ilmenee jo siirtoja, tehtävien vaihtamista, uudelleenopettelua kuin osittain tehtyä työtä. Näiden aiheuttamaa hukkaa ei kuitenkaan saatu mitattua työpäivissä. Oletettavasti nämä myös ilmenivät ensimmäisen version valmistumisen jälkeen, kun käyttötapaus siirtyi määrittelijältä toiselle.

Hukkaa ilmeni, kun valmistumisensa jälkeen käyttötapaus odotti toteutuksen aloitusta 162 työpäivää. Kuten aliluvussa 5.3.1. esitellyn käyttötapauksen 1 kohdalla, tässä ilmenee niin osittain tehtyä työtä, viivästystä kuin uudelleenopetteluakin. Käyttötapauksen toteutukseen osallistuneen kehittäjän mukaan toteutuksen aikana tekijät vaihtuivat usein ja käyttötapauksen määritykset muuttuivat jatkuvasti tai olivat puutteellisia. Nämä seikat johtivat osittain tehtyyn työhön, siirtoihin, tehtävien vaihtamiseen sekä uudelleenopetteluun. Näistä ei saatu mitattua tarkkoja työpäivämääriä, mutta kehittäjän mukaan hukka oli merkittävää.

Kuten aiemmissa käyttötapauksissa, viat ja käyttöönoton odottelu aiheuttivat hukkaa. Kun toteutuksen viimeiset tehtävät oli suljettu, testaus-ja-korjaus veivät syklit vielä 111 työpäivää ja käyttöönottoa odotettiin 11 työpäivää.

Kaiken kaikkiaan kvantitatiivisesti suurimmat havainnot hukasta aiheuttivat viivytykset

ja viat. Kun lasketaan yhteen toteutuksen aloituksen odotuksesta (162 työpäivää), käyttöönoton odottelusta (11 työpäivää) sekä vikojen korjaamiseen käytetystä ajasta (110 työpäivää) johtuvat hukat, saadaan hukan osuudeksi käytetystä kokonaisajasta (635 työpäivää)

$$\frac{162 + 11 + 110}{635} \approx 0{,}44$$

eli yli 44 % ajasta oli hukkaa. Eroteltuna viivytysten osuus oli

$$\frac{162 + 11}{635} \approx 0{,}27$$

eli noin 27 % kokonaisajasta ja vikojen

$$\frac{110}{635} \approx 0{,}17$$

eli noin 17 % kokonaisajasta.

Tässä käyttötapauksessa sekä määrittelijä että kehittäjä toivat kuitenkin esiin useita muita hukan lähteitä, joista osa koettiin merkittäviksi, joten hukan osuus oli todennäköisesti tätä korkeampi. Arvovirran tehokkuus oli (toteutukseen käytetyt työpäivät lisättynä testaukseen käytetyt työpäivät jaettuna kokonaisajalla)

$$\frac{340 + 1}{635} \approx 0{,}53$$

eli noin 53 %. Hukkien yhteenveto on esitelty taulukossa 14.

Hukan tyyppi	**Millä tavalla havaittu**	**Mitattu hukka (jos laskettavissa) työpäivinä**
Osittain tehty työ	Haastatteluista	
Ylimääräiset ominaisuudet	Ei havaittu	
Uudelleenopettelu	Haastatteluista	
Siirrot	Haastatteluista	
Tehtävien vaihtaminen	Haastatteluista ja sähköisestä aineistosta	
Viivytykset	Haastatteluista ja sähköisestä aineistosta	162 + 11
Viat	Haastatteluista ja sähköisestä aineistosta	110

Taulukko 64: Käyttötapauksesta 3 mitattujen hukkien yhteenveto

5.3.4 Käyttötapauksissa havaittujen hukkien yhteenveto

Hukkaluokista vain ylimääräisiä ominaisuuksia ei havaittu missään käyttötapauksessa. Kaikista käyttötapauksista löydettiin osittain tehdystä työstä, vioista sekä viivytyksistä aiheutunutta hukkaa. Kahdesta tapauksesta kolmesta löydettiin myös uudelleenopettelua sekä tehtävien vaihtamista. Näistä viivytykset ja viat aiheuttivat selvästi eniten mitattavaa hukkaa käsittäen käyttötapauksiin käytetystä kokonaisajasta 43 – 61 %. Yhteenveto käyttötapauksien hukista on esitetty taulukossa 15.

Käyttötapauksen numero	**Havaitut hukat**	**Suurimmat hukat**	**Hukan määrä käyttötapaukseen käytetyistä työpäivistä**
1 (aliluku 5.3.1)	• Osittain tehty työ • Uudelleenopettelu • Tehtävien vaihtaminen • Viivytykset • Viat	Viivytykset ja viat	48 %
2 (aliluku 5.3.2)	• Osittain tehty työ • viivytykset • viat	Viivytykset ja viat	61 %
3 (aliluku 5.3.3)	• Osittain tehty työ • uudelleenopettelu • siirrot • tehtävien vaihtaminen • viivytykset • viat	Viivytykset ja viat	44 %

Taulukko 15: Yhteenveto käyttötapausten hukista

6 Analyysi

Tässä luvussa tehdään johtopäätöksiä luvussa 5 esiteltyjen tulosten pohjalta. Tarkoituksena on vastata aliluvun 1.1. taulukon 1 tutkimuskysymykseen 1

Mitkä olivat suurimmat hukkaa aiheuttaneet tekijät päättyneessä ohjelmistoprojektissa?

tutkimuskysymykseen 2

Millä toimilla voidaan havaittujen hukkatekijöiden vaikutusta projekteissa vähentää tai poistaa?

sekä apukysymykseen 2

Miten hukan lähteiden poistoa voidaan priorisoida?

ja antaa näiden vastausten kautta suosituksia sekä nostaa esiin mahdollisia kohteita jatkotutkimuksille.

6.1 *Johtopäätökset*

Tämän tutkielman päämääränä oli tehostaa ohjelmistokehitystoimiston ohjelmistokehitysprosessin arvovirtaa. Arvovirta on sitä tehokkaampi, mitä lyhempi sen tahtiaika on, eli mitä nopeammin prosessi tuottaa jotain arvokasta. Asiakkaan näkökulmasta taas läpimenoaika kokonaisuudessaan on tärkeä, sillä mitä nopeampi läpimenoaika on, sitä nopeammin asiakas saa arvoa omaan käyttöönsä. Kumpaakin aikaa voidaan lyhentää poistamalla prosessien työvaiheista hukkaa.

Hukkaa, ei kaikissa tilanteissa, voi poistaa kaikista prosessin vaiheista samaan aikaan. Tästä syystä tässä tutkielmassa yhtenä pyrkimyksenä on priorisoida hukan poiston kohteita, mittaamalla syntyneen hukan määrää. Tutkimuksen kohteena olleesta projektista hukkaa oli pintapuolisesti helppo havaita ja kaikista tarkastelluista käyttötapauksista havaittiin monen tyyppistä hukkaa. Ainoastaan ylimääräisistä ominaisuuksista johtuvaa hukkaa ei löydetty, mutta tästäkin löytyi viite laadunvarmistustapaamisen muistiinpanoissa.

Hukan määrien tai niiden vaikutusten mittaaminen järjestelmästä, jota ei sitä varten ole erikseen valmisteltu, osoittautui kuitenkin haastavaksi. Projektin sähköiset työtilat mahdollistivat työtehtävien sekä niiden keston tarkan seurannan, mutta järjestelmää käytettiin tutkitussa projektissa vajavaisesti, ilmeisesti koska tarkemmalle seurannalle ei koet-

tu tarvetta. Näin ollen, vaikka monista hukista saatiin vahvoja viitteitä, vain vioista ja viivytyksistä johtuvaa hukkaa voitiin kvantitatiivisesti mitata. Käyttötapausten läpimenoajat olivat melko identtisiä käyttötapauksen arvioidusta koosta tai monimutkaisuudesta huolimatta (käyttötapaus 1: 622 työpäivää, käyttötapaus 2: 624 työpäivää, käyttötapaus 3: 635 työpäivää). Kuten aliluvun 5.4.3. taulukosta 16 nähdään, näistä läpimenoajoista merkittävä osa oli viivytyksistä ja odottelusta johtuvaa hukkaa: käyttötapauksen 1 kohdalla 48 %, käyttötapauksen 2 kohdalla 61 % ja käyttötapauksen 3 kohdalla 44 % kokonaisajasta.

Iso osa projektista mitatusta viivytyksistä oli käyttötapausten valmistumisen ja sen toteutuksen aloittamisen välistä aikaa. Itse odotusaika selittyy sillä, että kaikki projektin käyttötapaukset kirjoitettiin valmiiksi, ennen kuin yhtäkään niistä otettiin toteutukseen. Näin ollen viimeisenä valmistunut käyttötapaus toimi kaikkien käyttötapausten läpimenoaikaa rajaavana tekijänä. Niiden valmistumista taas rajoitti kirjoittamiseen tarvittavan tiedon ja resurssien saatavuus.

Kuten aliluvussa 5.3.3. olevasta käyttötapauksen 3 haastattelusta ilmenee, käyttötapausten kirjoittamiseen tarvittava tieto oli usein tiivistynyt yhteen henkilöön. Koska käyttötapauksia oli huomattavasti enemmän kuin niiden kirjoittajia, teki yksittäinen kirjoittaja useita käyttötapauksia. Lisäksi käyttötapausten 1 ja 3 haastatteluissa myös mainitaan, että kirjoittajat tekivät niitä usein päällekkäin. Näistä syistä johtuen henkilö, jolle tarvittava tieto oli kertynyt, ei pystynyt tehokkaasti suorittamaan omia tehtäviään tietopyynnöistä johtuvien jatkuvien keskeytysten takia ja muut henkilöt joutuivat odottamaan tarvittavia tietoja.

Tämä näkyi myös koettuun hukkaan liittyvän kyselyn vastauksissa, jotka käsiteltiin aliluvussa 5.1.1.. Kun kyselyn tuloksia tarkasteltiin yleisellä tasolla, viivytyksiä koettiin esiintyvän projekteissa kohtalaisia määriä. Lähempi tarkastelu paljasti kuitenkin, että liiketoiminnan edustajat kokivat erityisesti resurssien odottelun sekä tarvittavan tiedon vaikean saatavuuden aiheuttavan paljon viivytyksiä. Myös kehittäjät kokivat vastaavia viivytyksiä kohtalaisesti, mutta kehittäjät pystyivät itsenäisemmin jatkamaan työskentelyä jonkin toisen tehtävän parissa. Tämä todennäköisesti vaikutti siihen, kuinka vahvasti viivytykset koettiin.

Kuten luvussa 2 mura- muri- ja muda-termien esittelyssä mainittiin, hukkien syntyprosessit ovat usein sidoksissa toisiinsa ja näin ollen hukat usein esiintyvät yhdessä. Kohdeprojektin kohdalla käyttötapausten toteutuksen aloituksen odottelu aiheutti puhtaan

viivytyksen lisäksi osittain tehdyn työn hukkaa, jatkokäsittelyä odottavan valmiin käyttötapauksen muodossa. Kuten aliluvussa 5.2. esitellyistä laadunvarmistustapaamisen muistiinpanoista sekä aliluvussa 5.3. olevista käyttötapauksiin liittyvissä haastatteluissa ilmenee, monista käyttötapauksista paljastui merkittäviä ongelmia toteutuksen aikana ja usein niitä jouduttiin muuttamaan kesken toteutuksen. Tämä johti vikoihin, kun toteutuksen yksityiskohdat olivat epäselvät, ylimääräisiin viivästyksiin sekä tehtävien vaihtamiseen, kun toteutustyö jäi odottamaan muutoksien valmistumista. Aliluvuissa 5.3.2. ja 5.3.3. läpikäytyjen käyttötapauksien 2 ja 3 kohdalla toteutuksen aloittamisen odottelu johti myös uudelleenopetteluun, sillä käyttötapaus 3 odotti aloitusta noin 10 kuukautta ja käyttötapaus 2 noin 6 kuukautta. Toteutuksen välitön aloittaminen käyttötapauksen valmistuttua ei tietenkään ole tae sille, etteikö käyttötapausta olisi tarvinnut muokata ja näin estää muiden ongelmien syntymistä. Nopea aloitus olisi kuitenkin todennäköisesti johtanut ongelmien tehokkaampaan ratkaisuun, kun tapaukseen liittyvät asiat ovat tuoreessa muistissa ja tarvittavat henkilöt olisivat olleet heti saatavilla. Kyselyn tulosten mukaan tuotteiden määritysten liian aikaista sulkemista koettiin tapahtuvan kohtalaisesti, niin kehittäjien, kuin liiketoiminnan edustajien mielestä.

Seuraavaksi suurin projektista mitatun viivytyksen lähde oli käyttöönoton odottelu. Tämä toimi myös läpimenoajan toisena rajaavana tekijänä, sillä kaikki toteutetut käyttötapaukset julkaistiin kerralla. Järjestelmän käyttöönottoon liittyi huomattava määrä koordinointia sekä koulutusta järjestelmää käyttävien vastapuolien kanssa. Järjestelmän käyttöönoton yhteydessä myös vanhoja järjestelmiä suljettiin ja niiden tietoja siirrettiin uuteen järjestelmään. Näin ollen ilman etukäteen sovittua käyttöönottopäivää olisi järjestelmän julkaisu ollut vaikeaa, joten tämä oli välttämätön, syntyneestä hukasta huolimatta.

Viivästykset olivat kvantitatiivisesti merkittävä hukan lähde (taulukko 15, käyttötapaus 1 42 %, käyttötapaus 2 28 %, käyttötapaus 3 27 % kokonaistyöajasta). Erityisesti käyttötapausten toteutuksen aloittamiseen liittyneen viivästyksen aiheuttamat lisäongelmat olivat kuitenkin työn laadun ja tehokkuuden kannalta vielä merkittävämpiä.

Käyttötapauksista mitattu vikojen aiheuttama hukka taas syntyi testaus-ja korjaussykleistä. Joidenkin käyttötapauksien toteutuksia testattiin jo toteutuksen aikana, mutta kaikkien tarkasteltujen käyttötapauksien kohdalla toteutuksiin liittyvien vikojen korjaus kesti pitkään varsinaisen toteutuksen valmistuttua. Tätä selittää osin odottelu, jota syntyi, kun testaajat odottivat korjauksien valmistumista. Tyypillisesti korjaukset tuotiin

testattaviksi isommissa erissä, sillä testaus tehtiin pääosin erillisessä testausympäristössä, jonne korjausten vieminen ei ollut triviaalia. Kuten aliluvun 5.2. laadunvarmistustapaamisen muistiinpanoista ilmenee, korjauksia jäi myös välillä pois testiympäristöön asennetuista versioista. Tämä johti ylimääräiseen analysointiin ja odotteluun, kun korjauksen tilaa yritettiin selvittää. Käänteisenä esimerkkinä käyttötapauksen 2 kohdalla valmiit korjaukset odottivat yhtä tiettyä testaajaa, jolla oli tarpeeksi tietotaitoa testitapauksien ajamiseen.

Suurin osa vikoihin käytetystä työajasta oli kuitenkin aidosti vikojen korjaamiseen käytettyä aikaa. Vikojen määrä itsessään on suhteellinen, eikä ilman vertailukohdetta ole merkityksellinen, mutta vikojen korjaamiseen käytettiin kuitenkin huomattavia määriä työaikaa (käyttötapaus 1: 6 %, käyttötapaus 2: 33 %, käyttötapaus 3: 17 % kokonaistyöajasta). Aliluvussa 5.1.1. läpikäydyistä kyselyn tuloksista taas on nähtävissä, että yleisesti vikoja koettiin esiintyvän paljon ja erityisesti liitetoiminnan edustajien mielestä vikoja esiintyi paljon. Tätä eroa voi osittain selittää se, että liiketoiminnan edustajat toimivat yleensä myös kehitetyn järjestelmän loppukäyttäjinä. Näin ollen järjestelmässä olevat viat vaikuttavat suoraan heidän työskentelyynsä ja siksi ne koetaan voimakkaammin. Vikojen aiheuttama hukka oli kuitenkin kvantitatiivisesti merkittävä ja, kuten viivytykset, johti myös muihin ongelmiin.

Tehtävien vaihtamisesta ja siirroista saatiin myös viitteitä. Käyttötapauksen 3 kehittäjä ja määrittelijä sanovat haastattelussa, että heillä oli yhtäaikaisesti useita työtehtäviä käynnissä ja he joutuivat hyppäämään tehtävästä toiseen. Myös tutkielman laatijan omat kokemukset kohdeprojektista tukevat tätä. Tämä korostui, jos työn alla oli jokin käyttötapauksen toteutus ja edellisen käyttötapauksen toteutuksen testaus oli kesken. Vikojen korjaus priorisoitiin aina uuden toteutustyön edelle, joten työ piti keskeyttää ja siirtyä vian korjaamisen pariin. Käyttötapausten 1 ja 2 kehittäjät tai määrittelijät eivät havainneet tämän kaltaisia ongelmia, käyttötapauksen 3 henkilöiden mukaan haitta ei ollut merkittävä. Kyselyn tulosten (aliluku 5.1.1.) mukaan tehtävien vaihtamista havaittiin kohtalaisesti, joten kokonaisuudessaan hukka ei ollut merkittävää.

Siirroista johtuvaa hukkaa oli havaittavissa lähinnä käyttötapauksessa 3, johon osallistui useita määrittelijöitä ja kehittäjiä. Siirrot aiheuttivat määrittelijöille jonkin verran uudelleenopettelua ja ylimääräistä analysointia, kun käyttötapauksen kirjoittamisvastuu siirtyi tekijältä toiselle. Haastatellun kehittäjän mukaan toteutukseen osallistui useita määräaikaisia konsultteja ja tämän takia siirtoja tapahtui paljon. Siirtojen yhteydessä tietoa huk-

kui jatkuvasti ja joitain toteutuksen osia jouduttiin tekemään uudestaan tämän takia. Haitta oli siis merkittävä. Hukan määrää ei kuitenkaan voitu mitata, sitä ei havaittu muissa käyttötapauksissa ja kyselyn mukaan siirtoja esiintyi kohtalaisesti. Näin ollen hukan ei voida sanoa olevan merkittävä.

Kaiken kaikkiaan tulosten perusteella viivytykset aiheuttivat merkittävimmän määrän hukkaa kohdeprojektissa. Kyselyn mukaan viivytyksiä havaittiin kuitenkin vain kohtalaisesti, tosin liiketoimintapuolella korostui resurssien odottelusta ja tiedon vaikeasta saatavuudesta johtuvat viiveet. Seuraavaksi eniten hukkaa aiheuttivat viat, kyselyn mukaan näitä havaittiin paljon. Myös muita hukkia havaittiin, mutta näiden määrää ei saatu mitattua ja kyselyn tulosten mukaan niitä ei havaittu erityisen paljon.

Näin ollen vastauksena taulukossa 1 esiteltyyn tutkimuskysymykseen 1 **”Mitkä olivat suurimmat hukka aiheuttaneet tekijät päättyneessä ohjelmistoprojektissa”**, suurimmat hukkatekijät olivat viivytykset ja viat. Vastaus tutkimuksen apukysymykseen 2 **”Miten hukan lähteiden poistoa voidaan priorisoida?”** saadaan yhdistämällä eri hukkien mittaamalla havaitut määrät niiden koettuihin määriin. Mikäli hukkaa on mitattu merkittäviä määriä, ja sitä koetaan esiintyvän paljon, tulee se priorisoida korkealle poistettavien listalla. Tästä syystä ohjelmistokehitysprosessien tulisi priorisoida viivytyksistä ja vioista johtuvien hukkien poistoa.

6.2 *Suositukset*

Tässä aliluvussa vastataan aliluvun 1.1. taulukon 1 tutkimuskysymykseen 2 **”Millä toimilla voidaan havaittujen hukkatekijöiden vaikutusta vähentää?”,** antamalla suosituksia hukan poistoon. Ensimmäinen parannuskohde viivytyksistä johtuvan hukan poistamiseen on vaatimusmäärittelyjen valmistumisen ja niiden toteutukseen ottamisen välisen ajan lyhentäminen. Ohjelmistokehitysprosessit tulisi järjestää niin, että valmistunut vaatimusmäärittely voidaan ottaa heti toteutukseen. Toteutus tulisi myös pystyä rakentamaan yhtäjaksoisesti valmiiksi tai ainakin sellaiseen pisteeseen, jossa sitä tai sen osaa voidaan mielekkäästi hyväksymistestata. Näin määritysten tai toteutuksen ongelmat havaitaan varhain ja niihin voidaan reagoida nopeasti. Tällä toimintatavalla voidaan myös vähentää työtehtävien vaihtelua ja uudelleenopettelua, kun henkilöstö keskittyy saamaan yhden ominaisuuden kerrallaan valmiiksi, eikä joudu siirtymään kontekstista toiseen jatkuvasti.

Toteutetut ominaisuudet olisi suositeltavaa ottaa mahdollisimman nopeasti käyttöön.

Kaikissa tapauksissa ominaisuuksia ei kuitenkaan voida erinäisistä syistä johtuen käyttöönottaa yksittäin tai pienissä erissä, vaan ne julkaistaan kerralla osana isoa kokonaisuutta. Vaikka ominaisuudet joutuisivat odottamaan virallista käyttöönottoa, voidaan ne kuitenkin testata ja hyväksyä sisäisesti, mielellään välittömästi toteutuksen valmistumisen jälkeen.

Kohdeprojektin loppupuolella otettiin myös käyttöön tapa, jossa jonkin aikaa toteutuksen aloittamisen jälkeen määrittelijä sekä toteuttaja istuivat yhdessä alas ja kävivät läpi toteutusta sekä määritystä. Tällä tavoin saatiin tehokkaasti sovittua yhteinen näkemys siitä, miltä valmis toteutus tulisi näyttämään ja tämä johti toteutuksen laadun paranemiseen. Tapa todettiin hyväksi käytännöksi niin määrittelijöiden, kuin kehittäjienkin mielestä ja se suositellaan otettavaksi käyttöön kaikissa projekteissa.

Odottelua muodostui myös kehitystyöhön tarvittavan tiedon vaikeasta saatavuudesta. Tämä johtui suurelta osin tiedon kasaantumisesta muutamalle henkilölle. Näille samoille henkilöille kerääntyi myös muita enemmän vastuita ja tehtäviä, joten monessa tapauksessa heidän apuaan piti odottaa jonkin aikaa. Tiedon tehokas jakaminen on tunnetusti haasteellista, eikä tämän kaltaisten ongelmien ratkaisemiseen ole yhtä ainoaa oikeaa ratkaisua, mutta tulevissa projekteissa tähän kannattaa kiinnittää erityistä huomiota ja pyrkiä järjestelmällisesti parantamaan tiedon saatavuutta. Myös Lean-periaatteiden mukaisesti työkuormaa tulisi jakaa tasaisesti ja mieluummin ali- kuin ylikuormittaa resursseja.

Vikoja ei voida täysin estää syntymästä. Vikojen korjaamista voidaan kuitenkin tehostaa testauksen automatisoinnilla sekä jatkuvalla integroinnilla. Kattavilla testeillä vikatilanteet havaitaan nopeasti ja niihin voidaan näin ollen reagoida nopeasti. Näin käyttäjätestaukseen siirtyy vähemmän virheitä, ja vältytään aikaa vieviltä testaus- ja korjaussykleiltä. Tutkitun projektin loppupuolella näitä menetelmiä otettiin käyttöön, ja alkuvaikeuksien jälkeen niiden koettiin parantavan järjestelmän laatua.

Prosessi, jolla korjauksia hoidettiin, saattaisi olla myös hyvä tehostamisen kohde. Korjaukset olivat aina korkealla prioriteetilla ja menivät muun toteutustyön edelle. Näin ollen kehittäjät joutuivat usein keskeyttämään meneillään olevan työtehtävän ja syventymään korjauksen kohteena olevaan komponenttiin. Korjauksen jälkeen kehittäjän on taas palautettava mieleen aikaisempaan tehtävään liittyvät asiat. Toimivampi vaihtoehto voisi olla erillisen virhekorjausjakson varaaminen esimerkiksi sprintin loppupuolelle. Näin toteutustehtävät voitaisiin tehdä rauhassa loppuun ja kontekstista toiseen siirtymis-

tä vaativiin virhekorjauksiin voitaisiin varautua paremmin.

6.3 *Tutkimuksen rajoitukset*

Kuten Robson [Rob02] toteaa, yksittäisistä tapaustutkimuksista on yleensä hankalaa tehdä laajoja yleistyksiä. Hukkaan johtavat toimintatavat ovat vahvasti sidottu organisaation kulttuuriin, ja kuten missä tahansa ihmiskulttuurissa, niissä on valtavasti variaatiota. Koska tässä tutkielmassa tutkimus rajattiin yhteen organisaatioon ja sen yhteen projektiin, tuloksia ei voida suoraan yleistää tutkitun organisaation ulkopuolella.

Kerätyn kvantitatiivisen tiedon luotettavuuteen ja edustavuuteen liittyy muutama huomio. Kuten johtopäätöksien yhteydessä mainittiin, arvovirtojen määrittäminen järjestelmästä, jota ei sitä varten ollut erikseen valmisteltu, oli haasteellista. Kaikkia käyttötapauksien käsittelyyn liittyviä työvaiheita ei merkitty sähköisiin työtiloihin. Niistä työtehtävistä, jotka oli merkitty, ei aina ollut merkitty kaikkia vaiheita. Kun vaiheet olivat merkitty, niitä ei aina muistettu merkitä oikeaan aikaan, vaan esimerkiksi jokin tehtävä saatettiin merkitä valmiiksi vasta viikko sen jälkeen, kun tehtävä oli oikeasti valmistunut. Näin ollen aidot hukkien määrät saattavat poiketa mitatuista määristä muutamia päiviä. Tästä syystä myös monen hukan määrää ei saatu mitattua. Haastatteluiden ja kyselyn perusteella mittaamatta jääneet hukat eivät kuitenkaan olleet merkittäviä.

Laadunvarmistustapaamisen muistiinpanoissa esiintyi maininta käyttämättä jätetyn työntekijöiden luovuudesta johtuvasta hukasta. Hukkaa ei käsitelty, koska sille ei löytynyt vastaavaa luokkaa Poppendieckien aliluvussa 2.1. esitellyistä ohjelmistokehitykseen sovelletuista hukista. Siitä oli myös vain yksi maininta, eikä vastaavia mainintoja löytynyt muusta aineistosta. Näin ollen hukan ei voida sanoa olleen merkittävää. Vaikkakin hukan käsittelemättä jättäminen sinällään heikentää tulosten tarkkuutta, olisi hukan sisällyttäminen vaatinut teorian soveltamista tavalla, joka olisi saattanut johtaa muun aineiston vertailukelpoisuuden heikentymiseen. Näin ollen tutkielman laatija koki parhaaksi jättää hukan käsittelemättä.

Kuten aliluvussa 4.3.1. mainittiin, analysoinnin mahdollistavia käyttötapauskokonaisuuksia oli 102. Mitatun hukan tuloksien tarkkuutta olisi voitu edelleen parantaa tutkimalla nämä kaikki ja laskemalla niiden tietojen avulla keskiarvoja arvovirroille, yksittäisten esimerkkitapausten sijaan. Esimerkkitapaukset olivat kuitenkin edustavia ja niistä saadut havainnot olivat linjassa kyselyjen sekä haastatteluiden kokemusten kanssa.

Kyselyistä ja haastatteluista saatu kvalitatiivinen tieto oli subjektiivista. Haastatteluissa

tapa, jolla kysymykset esitettiin, vaihteli ja keskustelun yleinen vapaamuotoisuus saattoi myös vaikuttaa vastauksiin. Haastattelusta saadun tiedon laatua olisi voitu parantaa, mikäli samaan tapaukseen liittyen olisi voitu haastatella useampia henkilöitä. Tutkielman yhtenä tavoitteena oli kuitenkin hakea nimenomaan subjektiivisia kokemuksia hukasta ja saadut tiedot olivat samankaltaisia kvantitatiivisen tiedon kanssa.

Kyselyn tiedon laatua olisi voitu parantaa lisäämällä määrällisen valinta-asteikon rinnalle kriittisyyttä kuvaava asteikko. Toteutetussa kyselyssä vastattiin lähinnä siihen, kuinka paljon tai usein jotain hukkaa esiintyy. Jos vastaaja olisi vielä merkinnyt, kuinka vaikuttavaksi tai häiritseväksi kyseisen hukan kokee, olisi tätä tietoa voitu käyttää priorisoinnin tarkentamisessa.

Tulosten luotettavuuteen vaikuttaa myös se, että tutkielman laatija työskenteli itse osana tutkimuksen kohteena olevaa projektia. Toisaalta tämä mahdollisti havaintojen tekemisen toiminnan keskeltä ja antoi paremman ymmärryksen tiedoissa olevista aukoista, mutta toisaalta tämä saattoi vääristää tulosten tulkintaa.

Näitä heikkouksia kuitenkin korjattiin tiedon ja menetelmien trianguloinnilla. Kvalitatiivista ja kvantitatiivista tietoa kerättiin tietojärjestelmistä, muistiinpanoista ja haastatteluista useilta eri henkilöiltä. Nämä henkilöt myös edustivat erilaisia rooleja ja näin ollen erilaisia näkökulmia tutkittuun aiheeseen.

Tapausorganisaation näkökulmasta tulokset ovat luotettavia ja niitä voidaan käyttää parantamaan ohjelmistotuotannon toimintaa. Vaikka ne eivät ole helposti yleistettävissä, hukan havaitsemiseen käytetyt menetelmät ovat linjassa aliluvussa 2.5. esiteltyjen aiempien tutkimusten kanssa ja voidaan näin ollen soveltaa muihinkin tapauksiin. Myös hukan priorisointia havaitun määrän ja kokemusten mukaan voidaan käyttää muissa konteksteissa.

6.3.1 Validiteetti Runeson ja Höstin suositusten valossa

Runeson ja Höst [RuH09] ehdottavat tarkistuslistaa, jonka avulla tapaustutkimuksen validiteettiä voidaan tarkastella. Tässä aliluvussa tarkastellaan tätä tutkielmaa tämän valossa ja vastataan eksplisiittisesti tarkastuslistan kysymyksiin.

Tapaustutkimuksen suunnitelma

1. *Mikä on tutkittava tapaus?*

   Tapaus on päättynyt ohjelmistoprojekti.

2. *Onko tutkimuksen tavoitteet, alustavat kysymykset ja hypoteesit määritelty ennalta?*

   On. Tutkimuskysymykset ja tutkimuksen tavoitteet on määritelty johdantokappaleessa.

3. *Onko teoriatausta ja suhde olemassa olevaan kirjallisuuteen sekä aikaisempaan tutkimukseen määritelty?*

   On. Teoriatausta esiteltiin luvuissa 2, 3 sekä 4. Aiempia tutkimuksia esiteltiin aliluvussa 2.5

4. *Onko tutkielman laatijan tavoitteet tutkimuksen suhteen esitelty selvästi?*

   On. Tutkimuksen tavoitteet ovat yksiselitteisesti esitelty aliluvussa 1.1.

5. *Onko tapaus selvästi määritelty (koko, aihealue, prosessi, kohteet…)?*

   On. Tutkimus on selvästi rajattu yhden organisaation yhteen projektiin. Tutkimusta tukevaa aineistoa tosin kerätty myös projektin ulkopuolelta, projektia toteuttavan organisaation tasolta.

6. *Tutkitaanko syy-seuraussuhdetta? Jos tutkitaan, voidaanko ehdotetulla tutkimuksella erotella syy muista vaikuttavista tekijöistä?*

   Tutkielmassa tarkastellaan miksi ja miten hukkaa syntyy, eli tämän voidaan katsoa olevan syy-seuraussuhteen tarkastelua. Hukat ja niiden syntymekanismit ovat hyvin dynaamisia ja osittain päällekkäisiä, eikä tutkielmassa tarkastella yksittäisiä hukkien niin tarkalla tasolla, että kaikkia niihin vaikuttavia tekijöitä voitaisiin havaita. Näin ollen vastaus on, että eri vaikuttavia tekijöitä ei voida erotella.

7. *Liittyykö tutkimussuunnitelmaan tiedonkeräystä useasta lähteestä (tiedon triangulointi) usealla eri metodilla (menetelmien triangulointi)?*

   Tietoa kerättiin sähköisistä järjestelmistä, haastatteluista ja kyselyllä useilta eri henkilöiltä. Tiedon keräystä ja lähteitä esiteltiin luvussa 4.

8. *Onko syyt kohteiden, artefaktien, roolien, näkökulmien, jne. valintojen taustalla selkeät?*

   Syyt valinnoille on esitelty, esimerkkinä käyttötapausten valinta aliluvussa 4.2.1.

9. *Onko valittu tapaus relevantti vastamaan tutkimuskysymyksiin (rakenteellinen*

validius)?

Kyllä. Tapaus on ohjelmistokehitysprojekti, jossa esiintyy hukkaa monessa muodossa.

10. Onko yksilöiden ja organisaatioiden integriteetti otettu huomioon?

Aliluvussa 6.3. on yhdeksi tutkimuksen validiteettia rajoittavaksi tekijäksi mainittu tutkielman laatijan osallistuminen tutkimuksen kohteena olevaan projektiin. Muiden yksilöiden ja organisaation integriteettiä ei ole käsitelty.

Tiedonkeräykseen valmistautuminen

11. Onko tapaustutkimuksen protokollat tiedon keräämiseen ja analyysiin johdettu muista lähteistä? Onko toimenpiteet niiden päivittämiseen määritelty?

Protokollat, jotka on esitelty aliluvussa 4.1., on hyvin tunnettuja ja valittu olemassa olevan kirjallisuuden perusteella. Päivitystoimenpiteitä ei ole esitelty.

12. Onko tiedon keräyksessä suunniteltu käytettäväksi useita lähteitä ja keräysmenetelmiä?

On. Katso vastaus kysymykseen 7.

13. Onko mittausvälineet sekä menetelmät selvästi määritelty?

On. Mittauskohteet ja mittaustavat on esitelty aliluvussa 4.2. sekä liitteessä 2.

14. Ovatko suunnitellut menetelmät sekä mittarit riittävät täyttääkseen tutkimuksen tavoitteet?

Kyllä. Hukka voitiin havaita ja sen poistoa priorisoida saaduilla tuloksilla. Aliluvussa 6.3. käsitellään mittauksissa havaittuja puutteita.

15. Onko suunnitelma hyväksytty asiantuntijapaneelin toimesta ja onko tutkimuksen suorittamiseen haettu lupaa yksilöiltä ja organisaatioilta?

Suunnitelma on hyväksytty kohdeorganisaation sekä toteuttajaorganisaation puolesta.

Tiedon keräys

16. Onko tieto kerätty suunnitellun protokollan mukaisesti?

On. Keräyksen suunnitelma on esitelty luvussa 4, kerätty tieto luvussa 5.

17. Onko tutkittava ilmiö implementoitu oikein (missä määrin sitä oikeasti käyte-

tään)?

Tutkittu projekti on edustava esimerkki sen toteuttaneen organisaation ohjelmistokehitysprojekteista.

18. Onko tieto tallennettu jatkoanalyysien mahdollistamiseksi?

Suurin osa tiedosta on tallennettu sähköisessä muodossa, vain tutkielman laatijan omat havainnot sekä jotkin haastatteluvastaukset eivät ole tallessa.

19. Onko arkaluonteinen tieto tunnistettu tuloksista?

On. Kohdeorganisaation edustaja on tarkistanut tutkielman, ja salaiseksi tai arkaluonteiseksi katsottu tieto on poistettu.

20. Ovatko tiedon keräyksen menetelmät helposti toistettavissa?

Osittain kyllä. Kyselyt ja haastattelut voidaan toteuttaa sellaisinaan tapausorganisaation ulkopuolella, mutta sähköiset järjestelmät, joista kvantitatiivinen tieto kerättiin, ovat ainutlaatuisia tälle organisaatiolle. Järjestelmistä haluttava tieto on samaa kaikkialla, mutta yksityiskohdat siitä, miten tiedot kerätään, vaihtelevat.

21. Mahdollistaako kerätty tieto tutkimuskysymyksiin vastaamisen?

Kyllä. Tutkimuskysymyksiin on vastattu kerätyn tiedon perusteella luvussa 5.

Kerätyn tiedon analysointi

22. Onko analyysin menetelmistö selkeästi määriteltyä, mukaan lukien roolit ja katselmointimenetelmät?

Analyysin menetelmistö on esitelty aliluvussa 4.1., erillisiä rooleja ja katselmointimenetelmiä ei ole määritelty.

23. Onko kerätystä tiedosta esitetty selvä, jäljitettävissä oleva linkki tutkimuskysymyksiin ja olemassa olevaan tutkimustietoon?

Kyllä. Kuten kysymyksen 16 vastauksessa todettiin, tieto kerättiin protokollan mukaisesti, ja tämä protokolla perustuu olemassa olevaan tutkimustietoon. Tätä linkkiä käsiteltiin myös aliluvun 4.2. alussa.

24. Onko analyysissä tutkittu vaihtoehtoisia näkökulmia ja selityksiä?

Vain vähän.

25. Tutkitaanko syy-seuraussuhdetta? Jos tutkitaan, voidaanko ehdotetulla tutki-

muksella erotella syy muista vaikuttavista tekijöistä?

Katso vastaus kysymykseen 6.

26. *Tuottaako analyysi selkeitä päätelmiä, mukaan lukien suosituksia käytäntöjen muuttamiseksi ja jatkotutkimuskohteiksi?*

    Kyllä. Aliluvussa 6.1. vastataan eksplisiittisesti esitettyihin tutkimuskysymyksiin, aliluvussa 6.2. annetaan vastauksiin perustuen suosituksia ja aliluvussa 6.4 käydään läpi jatkotutkimuskohteita.

27. *Onko tutkimuksen validiteettiin kohdistuvat uhat analysoitu systemaattisesti ja onko vastatoimia implementoitu?*

    Joitain uhkia on tunnistettu ja analysoitu aliluvussa 6.3., mutta analyysi ei ole täysin kattava. Vastatoimia, kuten tiedon ja menetelmien triangulointia, on implementoitu mahdollisuuksien ja saatavilla olevien resurssien rajoissa.

Raportointi

28. *Onko tapaus ja siihen liittyvät analysoitavat yksiköt esitelty riittävän hyvin?*

    Kyllä. Tapaus ja siihen liittyvä organisaation on esitelty aliluvuissa 3.1. ja 3.3..

29. *Ovatko tavoitteet, tutkimuskysymykset ja niihin liittyvät vastaukset raportoitu?*

    Kyllä. Tavoitteet ja kysymykset esiteltiin luvussa 1, vastaukset luvussa 6.

30. *Onko tutkimukseen liittyvä teoria ja hypoteesit selkeästi raportoitu?*

    Tietoa teoriasta on kerätty useista lähteistä ja sitä on esitelty luvuissa 2 ja 4.

31. *Onko tiedon keräyksen menetelmät ja niiden valintaan liittyvät syyt esitelty?*

    Nämä on esitelty luvussa 4.

32. *Onko raakatietoa esitelty riittävästi?*

    Kaikki raakatieto mitä on voitu esitellä, on esitelty. Kyselyn tulokset ja laadunvarmistustapaamisen tiedot ovat tutkielman liitteenä. Haastattelut ovat audio tiedostoina, eikä niitä resurssien puutteen vuoksi voitu litteroida. Sähköisestä järjestelmästä kerätty raakatieto on arkaluonteista, eikä sitä voida näin ollen tutkielmaan sellaisenaan liittää.

33. *Onko analyysimenetelmät raportoitu selvästi?*

    Menetelmät on esitelty luvussa 4.

34. *Onko tutkimuksen validiteettiin kohdistuvat uhat ja niiden vastatoimet raportoitu selvästi?*

    Uhkia ja vastatoimia on käsitelty aliluvussa 6.3. Kirjoituksen selvyyteen tutkielman laatija itse on jäävi vastaamaan.

35. *Onko eettiset kysymykset raportoitu selvästi?*

    Kysymyksen 10 vastauksessa käsiteltiin yksilön ja organisaation integriteettiin liittyviä asioita, muuten eettisiä kysymyksiä ei ole käsitelty. Kirjoituksen selvyyteen tutkielman laatija itse on jäävi vastaamaan.

36. *Sisältääkö raportti päätelmiä, suosituksia käytäntöjen muutoksiksi ja jatkotutkimuskohteiksi?*

    Kyllä. Katso kysymyksen 26 vastaus.

37. *Antaako raportti realistisen ja uskottavan vaikutelman?*

    Aihetta on käsitelty tutkielman laatijan parhaan käsityksen ja osaamisen mukaan, noudattaen tapaustutkimukseen liittyvässä kirjallisuudessa esiintyviä hyväksi havaittuja käytäntöjä.

38. *Onko raportti sopiva kohdeyleisölle, helposti luettava ja hyvin jäsennelty?*

    Tähän kysymykseen tutkielman laatija on itse jäävi vastaamaan.

Tämä tutkielma täyttää valtaosan Runeson ja Höstin tarkistuslistan vaatimuksista. Näin ollen sen voidaan katsoa olevan niiden valossa validi.

6.4 *Jatkotutkimuskohteita*

Jatkotutkimusta voitaisiin tehdä tutkielmassa annettujen suositusten vaikuttavuudesta hukan poistamiseen ohjelmistokehitysprojekteissa. Tutkimukset voitaisiin rajata yksittäisiin tekijöihin. Esimerkiksi automaattisia testiajoja ja jatkuvaa integraatiota ei ole muissa organisaation projekteissa käytetty. Mikäli nämä implementoidaan jossain uudessa projektissa, voitaisiin virheiden määrää ja niihin käytettyä työaikaa verrata tässä tutkielmassa tutkittuun projektiin.

Mikäli työvaiheiden seurantaa parannetaan tulevissa projekteissa, olisi suositeltavaa kokonaisvaltaisesti tutkia niissä ilmenevää hukkaa. Seurannan parannusten myötä niissä ilmenevästä hukasta voitaisiin saada tarkempaa tietoa ja näin paremmin kohdistaa ongelmia korjaavia toimenpiteitä. Lisäksi mikäli seurantaa tehtäisiin standardinomaisesti, tutkimustiedon tuottamista voitaisiin automatisoida ja näin huomattavasti tehostaa mahdollisten ongelmien havaitsemista.

7 Yhteenveto

Tässä tutkielmassa päämääränä oli selvittää, kuinka tutkielman kohdeorganisaation ohjelmistokehitystä voidaan tehostaa Lean-menetelmillä. Tätä tarkoitusta varten tutkielmassa tutustuttiin aluksi Leanin perusperiaatteisiin ja sen soveltamiseen ohjelmistokehityksessä.

Leanin kaksi tunnetuinta ilmentymää ovat Toyota Production System (TPS) sekä Lean-ajattelu. TPS:n ydin periaatteita ovat: pitkän tähtäimen filosofia, oikeat prosessit, kehitä ihmisiä ja yhteistyökumppaneita sekä jatkuva oppiminen juuritason ongelmia ratkomalla. Lean-ajattelussa taas arvo, arvovirta, virtaus, imu ja täydellisyys ovat keskeisiä.

Poppendieckit [Pop03] sovelsivat Lean-ajattelua ohjelmistokehitykseen ja määrittelivät sille seitsemän pääperiaatetta: poista hukka, tee laadusta sisäsyntyistä, tehosta oppimista, lykkää päätöksentekoa, toimita nopeasti, kunnioita ihmisiä ja optimoi kokonaisuus. Kuten TPS:ssä, Lean-ohjelmistotuotannon päämääränä on tuottaa asiakkaalle mahdollisimman paljon arvoa mahdollisimman vähillä resursseilla. Lisäksi tuotantoprosesseja tehostetaan jatkuvasti poistamalla niistä kaikki arvoa tuottamaton toiminta, kuten ylimääräinen käsittely, turhat varastot ja tarpeeton odottelu, eli niin kutsuttua hukkaa.

Kohdeorganisaatiossa suoritettiin tapaustutkimus. Tutkimuksessa organisaation yhdestä merkittävästä ohjelmistokehitysprojektista havainnointiin hukkaa kvantitatiivisesti laskemalla läpimeno- ja tahtiaikoja eri työvaiheille. Tietoa hukasta kerättiin myös laadunvarmistustapaamisen muistiinpanoista sekä kyselyllä, jolla mitattiin subjektiivisesti koetun hukan määrää.

Suurimmat, niin mitatut, kuin koetut hukan lähteet olivat viivytykset sekä virheet. Näiden poistamiseksi suositeltiin kokonaisuuksien tekemistä kerralla loppuun saakka ilman keskeytyksiä sekä automaattitestauksen ja jatkuvan integroinnin käyttöön ottamista. Tulosten luotettavuutta vähensi projektin sähköisten järjestelmien epätarkkuus, tutkielman laatijan osallistuminen projektiin sekä tutkielman rajaaminen yhteen organisaation ja yhteen projektiin. Tulosten tarkkuutta pystyttiin kuitenkin parantamaan tiedon trianguloinnilla, ja tutkimuskohteena oleva organisaation pystyy hyödyntämään niitä oman toimintansa parantamisessa.

8 Lähdeluettelo

[DyD08] T. Dybå ja T. Dingsøyr, ”Empirical studies of agile software development: A systematic review,” *Information and Software Technology,* nro 50, pp. 833-589, 2008.

[CaG12] F. Calisir ja C. Gumussoy, ”Determinants of budget overruns on IT projects,” *Technovation,* nro 25, pp. 631-636, 2005.

[Had12] W. Hadid, ”The implementation of lean system in UK service industries,” tekijä: *Brunel Business School – Doctoral Symposium 27th & 28th March 2012,* London, 2012.

[WoJ98] J. P. Womack ja D. T. Jones, Lean Thinking, Lontoo: Productivity Press, 2003.

[Ram98] S. Raman, ”Lean Software Development: is it feasible?,” The Boeing Co, Seattle, 1998.

[Pop03] M. Poppendieck ja T. Poppendieck, Lean Software Development: An Agile Toolkit, Crawfordville, Indiana: Addison-Wesley Professional, 2003.

[Pop06] M. Poppendieck ja T. Poppendieck, Implementing Lean Software Development: From Concept to Cash, Stoughton, Massachusetts: Addison-Wesley Professional, 2006.

[BaG94] V. R. Basili, G. Caldiera ja R. D. H., ”The Goal Question Metric Approach,” 1994.

[Kra88] J. F. Krafcik, ”Triumph of the Lean Production System,” *Sloan Management Review,* nro 30, pp. 41-52, 1988.

[Lik04] J. K. Liker, The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer, McGraw-Hill, 2004.

[Mil56] G. A. Miller, ”The magical number seven, plus or minus two: Some limits on our capacity for processing information,” *Psychological Review,* nro 63, pp. 81-97, 1956.

[Kal09] T. Kalliomäki-Levanto, ”Keskeytykset ja katkokset työn etenemisessä,” TTL, Helsinki, 2009.

[DeL13] T. DeMarco ja T. Lister, Peopleware: Productive Projects and Teams, Addison-Wesley Professional, 2013.

[HED93] S. D. P. Harker, K. D. Eason ja J. E. Dobson, ”The change and evolution of requirements as a challenge to the practice of software engineering,” tekijä: *Proceedings of IEEE International Symposium on Requirements Engineering*, San Diego, CA, 1993.

[ClF91] Clark, Kim B. ja T. Fujimoto, "Product Development Performance: Strategy, Organization, and Management in the World Auto Industry", Harvard Business Press, 1991.

[IEE15] ”IEEE Xplore,” [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp. [Haettu 20 8 2015].

[Spr15] ”SpringerLink,” [Online]. Available: http://link.springer.com/. [Haettu 20 8 2015].

[Sci15] ”ScienceDirect,” [Online]. Available: http://www.sciencedirect.com/. [Haettu 20 8 2015].

[ACM15] ”ACM Digital Library,” [Online]. Available: http://dl.acm.org/. [Haettu 20 8 2015].

[RPK14] P. Rodriguez, J. Partanen, P. Kuvaja ja M. Oivo, ”Combining Lean Thinking and Agile Methods for Software Development,” tekijä: *47th Hawaii International Conference on System Science,* Waikoloa, 2014.

[MiJ12] P. Middleton ja D. Joyce, ”Lean Software Management: BBC Worldwide,” *IEEE Transactions On Eengineering Management,* osa/vuosik. 59, nro 1, pp. 20-32, 2012.

[TSR13] M. Tranco, J. Santos, J. L. Rodriquez ja J. Reich, ”Applying lean techniques to nougat fabrication: a seasonal,” *International Journal of Advanced Manufacturing Technology* , nro 68, pp. 1639-1654, 2013.

[PiM14] M. G. Pinheiro ja M. Misaghi, ”Proposal of a Framework of Lean Governance and Management of Enterprise IT,” tekijä: *Proceedings of iiWAS2014,* Hanoi, 2014.

[PeM10] K. Petersen ja C. Wohlin, ”Software process improvement through the Lean Measurement (SPI-LEAM) method,” *Journal of Systems and Software,* osa/vuosik. 83, nro 7, pp. 1275-1287, 2010.

[PoC14] K. Power ja K. Conboy, ”Impediments to Flow: Rethinking the Lean Concept of 'Waste' in Modern Software Development,” tekijä: *Agile Processes in Software Engineering and Extreme Programming,* Rome, 2014.

[IPO10] M. Ikonen, P. Kettunen, N. Oza ja P. Abrahamsson, ”Exploring the Sources of Waste in Kanban Software Developme,” tekijä: *P. in Proc. of SEAA Euromicro 2010* , Lille, 2010.

[MDP10] S. Mujtaba, R. Feldt ja K. Petersen, ”Waste and Lead Time Reduction in a Software Product Customization Process with Value Stream Maps,” tekijä: *21st Australian Software Engineering Conference,* Auckland, 2010.

[Pet12] K. Petersen, ”A Palette of Lean Indicators to Detect Waste in Software Maintenance: A Case Study,” tekijä: *Agile Processes in Software Engineering and Extreme Programming* , Berlin, Springer-Verlag, 2012, pp. 108-122.

[Coc02] A. Cockburn, ”Use cases, ten years later,” *STQE Magazine,* March/April 2002.

[RuH09] P. Runeson ja M. Höst, ”Guidelines for conducting and reporting case study,” *Empir Software Eng* , nro 14, pp. 131-164, 2009.

[Rob02] C. Robson, "Real World Research", Wiley, 2002.

[Yin02] R. K. Yin, "Case Study Research: Design and Methods", SAGE Publications, Inc, 2002.

[BGM87] I. Benbasat, D. K. Goldstein ja M. Mead, ”The case research strategy in studies of information systems,” *MIS Quarterly,* pp. 369-386, September 1987.

[SSS08] F. Shull, J. Singer ja D. I. K. Sjøberg, "Guide to Advanced Empirical Software Engineering", London: Springer, 2008.

[Giv08] L. M. Given, "The SAGE Encyclopedia of Qualitative Research Methods", London: SAGE Publication, Inc, 2008.

[Sea99] C. B. Seaman, ”Qualitative Methods in Empirical Studies,” *IEEE Transactions On Software Engineering,* osa/vuosik. 25, nro 4, pp. 557 - 572, 1999.

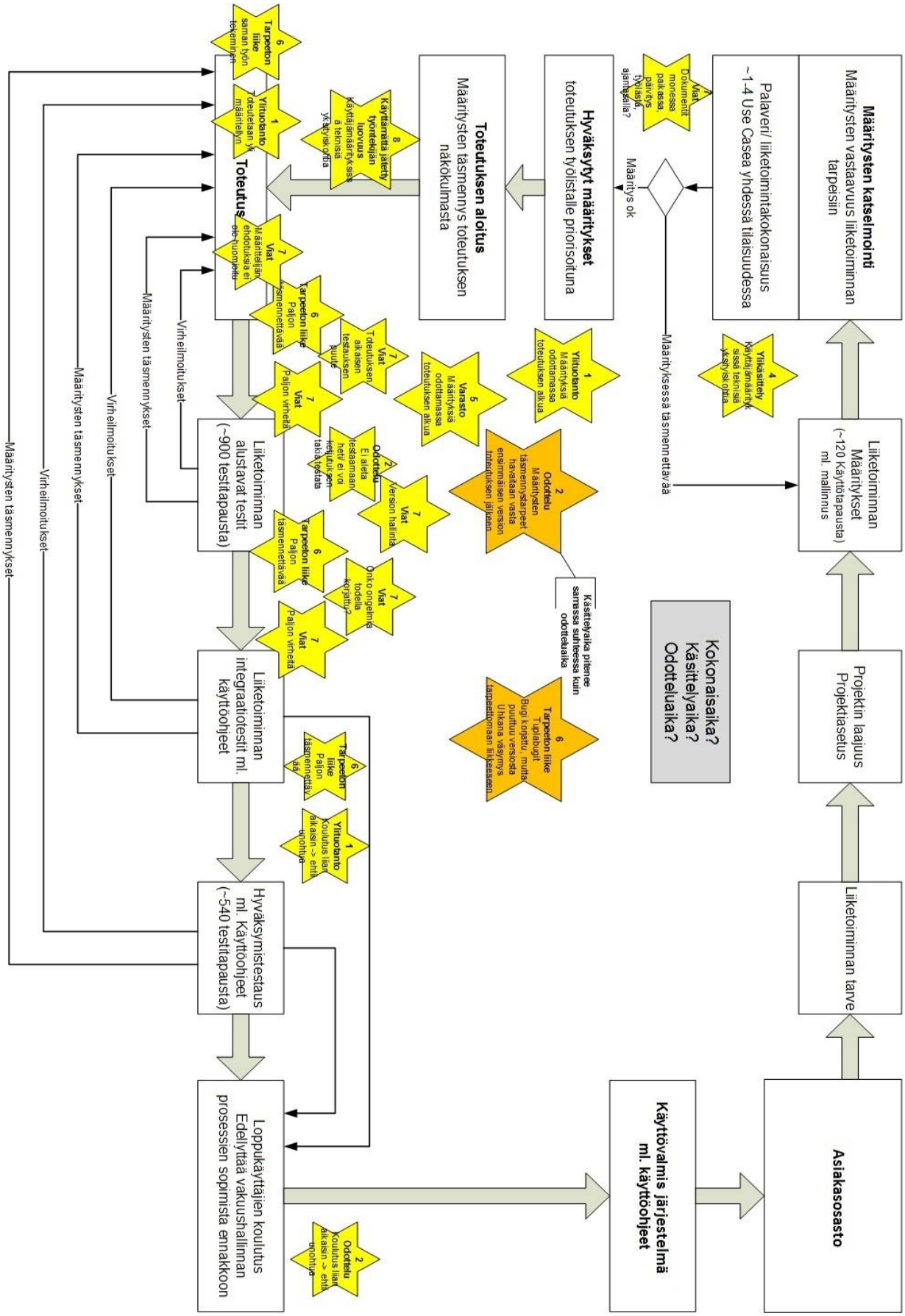
[Den10] M. Denscombe, "The Good Research Guide: for small-scale social research

projects", Open University Press, 2010.

[GMV12] D. Giesen, V. Merteens, R. Vis-Vicsschers ja D. Beukenhorst, "Questionnaire development 12", The Hague: Statistics Netherlands, 2012.

[JSB11] I. Jacobson, I. Spence ja K. Bittner, "Use-Case 2.0 The Guide to Succeeding with Use Cases", Ivar Jacobson International, 2011.

Liite 1: Laadunvarmistustapaamisen muistiinpanot



Liite 2: Koettu hukka- kyselyn kysymykset ja arvoasteikot

Ylimääräiset ominaisuudet (overproduction)

- *K1: Tuotteisiin/palveluihin määritellään ominaisuuksia, joita ei koskaan käytetä (1: Ei yhtään – 5: erittäin paljon)*
- *K2: Tuotteisiin/palveluihin määritellään ominaisuuksia, joita ei koskaan toteuta (1: Ei yhtään – 5: erittäin paljon)*
- *K3: Tuotteisiin toteutetaan ominaisuuksia, joita ei ole pyydetty (1: Ei yhtään – 5: erittäin paljon)*
- *K4: Tuotteisiin toteutetaan pyydettyjä ominaisuuksia, mutta ilman määrityksiä (1: Ei yhtään – 5: erittäin paljon)*

Viivytykset (waiting)

- *K5: Tuotekehityksen aikana aikaa kuluu resurssien odotteluun (1: Ei yhtään – 5: erittäin paljon)*
- *K6: Tuotekehityksen aikana aikaa kuluu tiedon odotteluun (1: Ei yhtään – 5: erittäin paljon)*
- *K7: Tuotekehityksen aikana aikaa kuluu päätösten odotteluun (1: Ei yhtään – 5: erittäin paljon)*
- *K8: Tuotekehitykseen tarvittava tieto on vaikeasti löydettävissä tai muuten vaikeasti saatavilla (1: Ei koskaan – 5: erittäin usein)*
- *K9: Tuotekehitykseen liittyen järjestetään turhia tapaamisia ja palavereita (1: Ei yhtään – 5: erittäin paljon)*
- *K10: Muutosten tekeminen vaatii liian monta hyväksyntää (1: Ei koskaan – 5: erittäin usein)*

Osittain tehty työ (inventory)

- *K11: Projektiin liittyvä työtehtävä keskeytyy pitkäksi aikaa (1: Ei yhtään – 5: erittäin paljon)*
- *K12: Tuotteiden/palveluiden määritykset lukitaan liian aikaisin (1: Ei koskaan –*

5: erittäin usein)

- *K13: On epäselvää, miksi tai mihin käyttöön tuotetta/palvelua kehitetään (1: Ei koskaan – 5: erittäin usein)*
- *K14: Tuotteita kehitetään epäluotettavan tiedon varassa (1: Ei yhtään – 5: erittäin paljon)*

Tehtävien vaihtaminen (motion)

- *K15: Projektiin/hankkeeseen liittyvä työtehtävä vaihtuu vaikka edellinen työtehtävä on vielä kesken (1: Ei koskaan – 5: erittäin usein)*
- *K16: Tuotteita/palveluita testataan liiallisesti (1: Ei koskaan – 5: erittäin usein)*

Uudelleenopettelu

- *K17: Keskeytykset työtehtävässä aiheuttavat uudelleenopettelua (1: Ei yhtään – 5: erittäin paljon)*
- *K18: Tuotekehityksessä hyväksikäytetään jo olemassa olevia malleja/määrityksiä/komponentteja (1: Ei yhtään – 5: erittäin paljon)*

Siirrot (transportation)

- *K19: Työtehtäviä siirretään henkilöltä toiselle (1: Ei yhtään – 5: erittäin paljon)*

Viat (defects)

- *K20: Valmiissa tuotteissa/palveluissa esiintyy vikoja (bugeja) (1: Ei yhtään – 5: erittäin paljon)*
- *K21: Epäsopivien/epäkypsien/tarpeettomien tekniikoiden ja teknologioiden käyttäminen aiheuttaa ongelmia (1: Ei koskaan – 5: erittäin usein)*

Liite 3: Koettu hukka- kyselyn tulokset, keskiarvot ja hajonnat

		K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18	K19	K20	K21
	Keskiarvo, IT	2,62	2,54	2,62	3,05	3,15	3,15	2,85	2,77	3,31	3,08	1,69	3,08	2,23	2,85	2,85	2,92	3,00	2,69	2,85	3,77	3,62
	Keskiarvo, Liiketoiminta	2,57	3,21	2,86	2,86	3,79	3,00	2,93	2,43	2,93	3,71	1,71	2,93	2,14	3,29	3,00	3,07	3,50	2,93	2,93	3,57	4,36
	Frekvenssi, IT																					
1		0	0	2	0	0	2	0	0	0	1	6	1	2	1	1	1	2	0	0	0	0
2		8	10	5	2	4	2	5	8	7	2	6	3	6	6	4	3	3	6	4	1	1
3		4	2	5	3	3	6	5	1	1	6	1	5	4	2	5	6	3	5	7	4	5
4		1	2	1	6	5	5	3	3	3	4	0	3	0	6	2	4	4	2	2	5	5
5		0	0	0	1	2	0	0	0	2	0	0	1	0	1	0	0	3	0	0	1	2
	Frekvenssi, Liiketoiminta																					
1		0	0	1	0	0	0	0	0	0	1	8	0	2	1	0	0	1	0	1	0	0
2		5	1	3	3	1	3	4	9	5	1	3	5	8	2	5	2	0	4	3	2	0
3		8	7	4	7	3	6	5	2	2	3	1	4	2	3	3	7	5	6	4	2	2
4		0	3	4	2	4	3	3	2	5	3	0	3	1	3	5	2	4	3	4	7	3
5		0	1	0	0	3	0	0	0	0	4	1	0	0	1	0	0	0	0	0	1	6
	Keskiarvo	2,54	2,85	2,64	3,21	3,56	3,12	2,88	2,52	3,00	3,32	1,65	2,92	2,16	3,04	2,84	2,96	3,20	2,81	2,88	3,57	3,96
	Arvojen summa	66,00	74,00	66,00	77,00	89,00	78,00	72,00	63,00	75,00	83,00	43,00	73,00	54,00	79,00	71,00	74,00	80,00	73,00	72,00	82,00	95,00
	Luokan keskiarvo	2,80				2,86							2,74				2,96	3,00		2,88	3,77	
	Luokan hajonta	0,89				1,09323							1,007006				0,7698	0,999314		0,785674	0,876652	
	Kysymykset luokittain																					
	Ylimääräiset ominaisuudet																					
K1	Tuotteisiin/palveluihin määritellään ominaisuuksia, jotka ei koskaan käytetä																					
K2	Tuotteisiin/palveluihin määritellään ominaisuuksia, jotka ei koskaan toteuta																					
K3	Tuotteisiin toteutetaan ominaisuuksia, jotka ei ole pyydetty																					
K4	Tuotteisiin toteutetaan pyydettyjä ominaisuuksia, mutta ilman määrityksiä																					
	Viivytykset																					
K5	Tuotekehityksen aikana aikaa kuluu resurssien odotteluun																					
K6	Tuotekehityksen aikana aikaa kuluu tiedon odotteluun																					
K7	Tuotekehityksen aikana aikaa kuluu päätösten odotteluun																					
K8	Muutosten tekeminen vaatii liian monta hyväksyntää																					
K9	Tuotekehityksen liittyen järjestetään turhia tapaamisia ja palavereita																					
K10	Tuotekehitykseen tarvittava tieto on valkeasti löydettävissä tai muuten vaikeasti saatavilla																					
K11	Tuotteita/palveluita testataan liallisesti																					
	Osittain tehty työ																					
K12	Tuotteiden/palveluiden määritykset lukitaan liian aikaisin																					
K13	On epäselvää, miksi tai mihin käyttöön tuotetta/palvelua kehitetään																					
K14	Projektiin liittyvä työtehtäviä keskeytyy pitkäksi aikaa																					
K15	Tuotteita kehitetään epäluotettavan tiedon varassa																					
	Tehtävien vaihtaminen																					
K16	Projektiin/hankkeeseen liittyvä työtehtäviä vaihtuu vaikka edellinen työtehtävä on vielä kesken																					
	Uudelleenopettelu																					
K17	Keskeytykset työtehtävässä aiheuttanut uudelleenopettelua																					
K18	Tuotekehityksessä hyväksikäytetään jo olemassaolevia malleja/määritykisiä/komponentteja																					
	Siirrot																					
K19	Työtehtäviä siirretään henkilöltä toiselle																					
	Viat																					
K20	Epäsopivien/epäkypsien/tarpeettomien tekniikoiden ja teknologioiden käyttäminen aiheuttaa ongelmia																					
K21	Valmiissa tuotteissa/palveluissa esiintyy vikoja (bugeja)																					

Liite 4: Koettu hukka- kyselyn tulokset, yksittäisten vastaajien tiedot

Rooli	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14	K15	K16	K17	K18	K19	K20	K21
IT	4	4	4	4	2	2	3	3	4	4	3	3	2	4	4	3	3	3	3	3	4
IT	2	2	4	5	3	3	3	4	5	5	1	5	3	2	4	4	5	2	3	5	5
IT	3	2	2	4	3	3	3	2	2	3	1	3	1	2	3	2	2	2	2	4	4
IT	3	4	3	4	3	4	3	4	4	3	1	3	3	4	4	4	3	3	4	5	3
IT	2	2	3	3	2	2	2	2	2	2	2	1	1	3	2	3	1	3	2	3	4
IT	2	2	1	2	3	3	4	2	4	4	1	4	2	2	2	3	3	2	2	4	4
IT	2	2	1	4	5	4	2	2	2	1	1	2	3	1	1	3	5	4	2	4	2
IT	2	2	2	4	2	3	2	2	5	3	2	2	2	2	2	1	1	4	3	2	3
IT	2	3	2	3	4	4	3	4	4	3	3	3	2	4	3	3	4	2	3	5	4
IT	3	2	3	5	2	2	2	2	5	3	1	5	2	4	3	4	5	2	3	3	4
IT	4	4	3	4	4	4	4	4	2	4	2	4	3	3	2	4	3	3	3	4	3
IT	3	2	4	4	4	3	2	3	2	3	2	3	3	2	3	2	2	3	3	3	4
IT	2	2	2	4	4	4	4	2	2	2	2	2	2	4	4	2	2	2	4	4	3
Liiketoiminta	2	2	2	2	2	3	3	2	2	4	2	4	2	2	3	3	4	3	3	3	3
Liiketoiminta	2	3	3	3	4	3	3	2	3	4	1	3	2	4	2	3	4	2	3	4	5
Liiketoiminta	3	3	2	3	2	3	2	2	2	1	1	2	1	4	2	4	4	3	3	2	3
Liiketoiminta	3	3	3	3	5	4	3	3	3	3	1	3	3	5	3	3	5	3	4	4	5
Liiketoiminta	2	4	4	3	4	4	4	4	4	5	5	2	2	4	4	4	4	4	4	4	5
Liiketoiminta	3	4	4	3	5	4	4	2	3	5	3	3	2	4	2	4	4	2	3	5	5
Liiketoiminta	3	3	2	3	3	3	2	3	4	4	2	4	3	1	2	2	3	2	2	4	3
Liiketoiminta	3	3	3	2	4	2	3	2	2	5	1	4	4	4	4	3	3	3	4	4	5
Liiketoiminta	3	5	4	4	4	3	3	2	2	3	2	2	2	3	4	3	4	2	2	4	5
Liiketoiminta	3	3	4	3	3	3	4	4	4	5	1	4	2	3	3	3	3	3	2	4	4
Liiketoiminta	2	3	3	3	4	3	3	2	4	4	1	3	2	5	4	3	4	3	3	3	5
Liiketoiminta	3	2	1	2	5	2	3	2	2	2	1	3	2	2	2	2	1	4	1	2	5
Liiketoiminta	2	4	2	4	5	3	2	2	4	4	1	2	1	2	4	3	3	4	4	4	4
Liiketoiminta	2	3	3	2	3	2	2	2	2	3	2	2	2	3	3	3	3	3	3	3	4
Keskihajonta koko	0,624159242	0,874889764	0,965807764	0,860662966	1,031732	0,716262	0,737027731	0,82817332	1,099944	1,130692	0,935506	0,981307	0,721985	1,11972	0,899627	0,7698	1,173554	0,721985	0,785674	0,860663	0,860663
Keskihajonta IT	0,737820234	0,842650088	1,002954216	0,769230769	0,948371	0,769231	0,769230769	0,89044899	1,263975	0,997037	0,721602	1,140954	0,696568	1,026282	0,948371	0,916644	1,358732	0,721602	0,661717	0,890449	0,73782
Keskihajonta Liiketoiminta	0,494871659	0,77261813	0,914732034	0,638876565	1,012675	0,654654	0,703489843	0,72843136	0,883523	1,160577	1,097307	0,798596	0,742307	1,160577	0,845154	0,59333	0,906327	0,70349	0,883523	0,820652	0,811273
Luokittain	0,885409858				1,09323							1,007006				0,7698	0,999314		0,785674	0,876652	